Widescale analysis of transcriptomics data using cloud computing methods

Anne M. Owen

A thesis submitted for the degree of
Doctor of Philosophy (Ph.D.)

Department of Mathematical Sciences

University of Essex

Date of submission: September 2015

Dedicated to

My husband Peter, our family and my supervisor

Acknowledgements

I am extremely grateful to my supervisor, Dr Andrew P. Harrison, for his inspiration and encouragement throughout the period of study for this degree. For many years I had wondered if I would be able to achieve the level of study required for a doctorate, having rejected the opportunities I was offered at age 23. Harry saw some potential and had faith in me at this later stage in my career. He has guided the research, read many chapter drafts and advised on the shape of the thesis.

My grateful thanks also go to Dr Adrian Clark of the Computing Science and Electrical Engineering Department of the University of Essex for backing up Harry as my second supervisor. Adrian enabled me to present to his Masters students about Cloud Computing and made very helpful comments on the draft thesis.

I would like to thank Dr Hugh P. Shanahan of Royal Holloway College, London University, for his collaboration and help during work on the Windows Azure Cloud, for his part in developing GWydiR and for his comments on reading some of the thesis chapters. My thanks also go to Prof. Graham Upton for his helpful statistical insights. Ibrahim Musa has been an inspiring collaborator in the work on a local cloud, and I am grateful for his enthusiastic support. Farhat Memon was a great encouragement as a fellow PhD student in Bioinformatics, and indeed it was while helping her to work on the Amazon cloud that the possibility of my study was suggested.

I am grateful for the R scripts used by Hugh Shanahan and Farhat Memon in their research into G-stacks as these formed the basis of the scripts used in the wide scale survey

work.

My husband, Peter, has been a tower of strength and support throughout my journey. He has made sacrifices in many ways. Above all I give glory to God who helped me start, progress and finish this thesis. He was my strength through many times of doubt and weakness.

Summary

This study explores the handling and analyzing of big data in the field of bioinformatics. The focus has been on improving the analysis of public domain data for Affymetrix GeneChips which are a widely used technology for measuring gene expression. Methods to determine the bias in gene expression due to G-stacks associated with runs of guanine in probes have been explored via the use of a grid and various types of cloud computing.

An attempt has been made to find the best way of storing and analyzing big data used in bioinformatics. A grid and various types of cloud computing have been employed. The experience gained in using a grid and different clouds has been reported. In the case of Windows Azure, a public cloud has been employed in a new way to demonstrate the use of the R statistical language for research in bioinformatics.

This work has studied the G-stack bias in a broad range of GeneChip data from public repositories. A wide scale survey has been carried out to determine the extent of the G-stack bias in four different chips across three different species. The study commenced with the human GeneChip HG_U133A. A second human GeneChip HG_U133_Plus2 was then examined, followed by a plant chip, Arabidopsis thaliana, and then a bacterium chip, Pseudomonas aeruginosa. Comparisons have also been made between the use of widely recognised algorithms RMA and PLIER for the normalization stage of extracting gene expression from GeneChip data.

Declaration

The study presented in this thesis is all my own work, except where otherwise stated and acknowledged. It has not already been accepted for any degree and is also not being concurrently submitted for any other degree.

Glossary

amino acids are biologically important organic compounds composed of amine (NH_2) and carboxylic (COOH) functional groups, together with a side-chain specific to each amino acid. Their key elements are carbon, hydrogen, oxygen and nitrogen. About 500 amino acids are known.

cytoplasm is a gel-like substance enclosed within a cell's membrane. Most cellular activities occur within the cytoplasm.

enzymes are highly selective catalysts which are responsible for the thousands of metabolic processes that sustain life. Most enzymes are proteins, though some catalytic RNA molecules have been identified.

G-stack is a sequence of four "G" (guanine) bases in a strand of nucleic acid such as RNA or DNA.

gene product is the biochemical material, either RNA or protein, which results from expression of a gene.

hybridization is the process of joining two complementary strands of nucleic acid sequences, e.g. RNA or DNA.

microarray is a 2D array on a glass slide for testing biological material using hibridization and laser scanning.

nucleic acids are large biological molecules which are essential for all known forms of life. DNA and RNA are examples. They encode, transmit and express genetic information.

nucleotides are the building blocks of nucleic acids like DNA and RNA. They are composed of a nitrogenous base with a five-carbon sugar and at least one phosphate group.

polymers are large molecules which are composed of many repeated subunits. Polymers range from familiar synthetic plastics like polystyrene to natural biopolymers like DNA and proteins.

photolithography is a process that uses light to control the manufacture of multiple layers of material.

polypeptides are long continuous chains of amino acids. In general they are smaller than proteins so have fewer amino acids (approximately 50 or fewer according to Wiki).

probe affinity (a description used by Irizarry *et al.* [1]) is the attraction which causes hybridization of strands of nucleic acid sequences. Abnormal probe affinity relates to a probe sequence having an affinity to bind to a neighbouring probe rather than to the desired RNA sequence. Affymetrix, to be strictly accurate, prefers to use the term "feature responses" in place of "probe affinities" due to the many factors which interact to produce measured intensity [2].

proteins consist of one or more polypeptides, arranged in a biologically functional way.

web role is a web application which is accessible via a *http://...* or *https://...* endpoint (webpage or web address)

worker role is a background processing application (program)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The study described in this thesis covers investigations made in the fields of bioinformatics and computer science. It is necessary to understand some biological and chemical structures in order to use bioinformatics and study the data that biologists generate so these structures have been briefly described. Computers are invaluable to perform the desired analyses and also to handle the large volumes of data that can be involved so there are chapters dedicated to explaining the type of computation facilities used.

Much use has been made in recent years of microarray technology to perform research into gene expression and the chemical processes behind cell development. Experiments are also creating vast amounts of sequencing data in a variety of fields where DNA and RNA are of interest. Microarray data has been examined here to establish methods of confirming the bias that can be introduced to this data by the particular probes chosen in the Affymetrix GeneChip® technology (see chapter 3). From the various types of bias that have been detected (for example by Langdon *et al.* [3] and by Upton and Harrison [4]) the effect of G-stacks (sequences of four or more "G"s) in the probes was chosen to be investigated in more detail. A wide scale survey of all experiments that used a particular human GeneChip has been carried out. This survey was repeated for another human GeneChip, and then for GeneChips in two other species.

An attempt has been made to find a good way of handling and analyzing big data in the

field of bioinformatics. The use of a grid and of cloud computing is explored to examine the process of investigating large amounts of data from microarray experiments. This type of data is often deposited and made available in public databases accessible across the internet. The use of a local private cloud has been demonstrated and in the case of Windows Azure, a public cloud has been employed in a new way to demonstrate the use of the R statistical language for research in bioinformatics.

Ever increasing quantities of data are being generated each year. This data can be analysed by other researchers from those who created it, in order, for example, to search for patterns of gene expression that might not have been anticipated. This work paves the way for fresh approaches in the handling and analysis of large volumes of biological data of any type.

1.1 Organisation of thesis

Chapter 2 introduces the basic principles of molecular biology in terms of the cell, DNA and RNA, genes and the genome, and bioinformatics and its tools. It concludes with an explanation of the challenge presented by "Big Data" in the field of bioinformatics, and with an overview of grid and cloud computing.

Chapter 3 explains the use of microarrays and the technology of Affymetrix GeneChips. It explores how gene expression is evaluated from the results of microarray experiments and how bias can be introduced by certain sequences such as G-stacks. This bias is the focus of further analysis described in later chapters. The generation of "unique mappings" is explored and analytic techniques of regression analysis and correlation matrices are explained as a way to evaluate the bias due to G-stacks.

Chapter 4 explores grid computing as a solution to the problem of accessing computer resources as a bioinformatician when fast processors and huge storage devices are not available locally. Grid computing is examined through experiences of using the NGS grid

which was the National Grid Service for the academic research community in the United Kingdom.

Chapter 5 contrasts the experiences of using a local private cloud and two public clouds: Amazon Web Services and Microsoft's Windows Azure. This research project enjoyed early access to both the Amazon and Azure cloud offerings, whose features and advantages have improved and grown enormously over the period.

Chapter 6 explains the experience of using Windows Azure cloud services when they were newly available, to begin to analyse a large quantity of microarray data. The development of bespoke interfaces to run R scripts in the Windows-centric environment of Azure is described. Data sets had to be uploaded from public repositories of microarray experiments. Finally some timings of running R scripts in the cloud is compared with timings on local computers.

Chapter 7 returns to the biological experiment data from microarrays to report on the experience of using the Azure cloud to analyse all the publicly available data from the human GeneChip HG_U133A. There are issues to be discussed when uploading large quantities of data from many experiments of different researchers. There was also the need to develop a new method of submitting multiple jobs to analyse the data. This work allowed the assessment of the extent of G-stack bias across a wide variety of experiments which used the HG_U133A chip.

Chapter 8 extends the wide scale analysis to another human GeneChip HG_U133_Plus and to two other species: the plant *Arabidopsis thaliana* and the bacterium *Pseudomonas aeruginosa*. Scatter plots are used to compare the results in each species between using RMA and PLIER, two different types of normalization routine. Expression data and correlation data across probe sets are both used to compare the effects of G-stacks in the probes of the microarray datasets of each species.

Chapter 9 summarises the conclusions of the thesis. It also suggests ways in which future research might profitably be directed.

1.2 Published papers

1. Musa, Ibrahim K., Owen, Anne M., Harrison, Andrew P. and Walker, Stuart D. (May 2014) Self-service infrastructure container for data intensive application. *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 3, no. 1

2. Shanahan, Hugh P., Owen, Anne M., Harrison, Andrew P. (Jan 2014) Bioinformatics on the cloud computing platform Azure. *PloS One 9.7* ISSN: 1932-6203. URL: http://dx.plos.org/10.1371/journal.pone.0102642

3. Shanahan, Hugh P., Owen, Anne M., Harrison, Andrew P. (July 2013) Integrating R with a Platform as a Service cloud computing platform for Bioinformatics applications. *The R User Conference, useR! 2013, Book of Contributed Abstracts* page 7 URL: http://www.edii.uclm.es/ useR-2013/abstracts/files/150_HughShanahanAzure.pdf

4. Memon, Farhat N., Owen, Anne M., Sanchez-Graillet, Olivia, Upton, Graham J.G., and Harrison, Andrew P. (2010) Identifying the impact of G-Quadruplexes on Affymetrix $3'$ Arrays using Cloud Computing. *Journal of Integrative Bioinformatics*, vol. 7, no. 2, 111

5. Memon, Farhat N., Sanchez-Graillet, Olivia, Upton, Graham J.G., Owen, Anne M., and Harrison, Andrew P. (2009) Identifying the impact of G-Quadruplexes on Affymetrix exon arrays using cloud computing. *CIB2009* URL: http://www.cls.zju.edu.cn/binfo/IB/2009/Proceeding.pdf#page=55.

1.3 My contribution

My contribution to the field of Bioinformatics through this thesis is the wide scale analysis of publicly available transcriptomics data, the results of which are given in detail in chapters 7 and 8. In order to achieve these results all possible microarray data from four different

GeneChips that was available in May 2012, was uploaded to a public cloud and analysed. The types of analysis chosen were based on work by Shanahan and Memon which is described in section 3.5. I modified their scripts to run on the Azure cloud.

Initially I was involved with my supervisor's team of researchers in creating unique mappings (see section 3.4.2) which identified probes that could be used as more reliable measures of target expression. Following this was the creation of heatmaps (see section 3.4.3.2) which visualize the correlation coefficients between pairs of probes in a probe set. The heatmaps showed that some probes were not correlated with other probes in their probe sets, and yet probes containing runs of guanine typically showed correlation with each other. The investigation of runs of guanine was showing them to be a cause of error or bias in measuring gene expression. I do not believe that there had ever been a wide scale survey done to find out the magnitude of this bias across all the available experimental data deposited in public databases.

Computing grids had been in use for several years but computing clouds were in their infancy, so with my background in computing science (M.Sc. Newcastle, 1973 as Anne Yates), I embarked on the use of a grid and cloud computing to perform the analysis runs. The experiences reported on using grid computing were with a script written by Farhat Memon. My experience on the Amazon cloud came initially from helping Farhat to get started with cloud computing. I was able to create some tailored machine images for her to use for our group research. I also uploaded and maintained data files in Amazon S3 storage.

My personal contribution to the work on Musa's local private cloud was firstly to supply the same data and R scripts that I was using to analyse the effect of runs of guanines. Then discussion of the bioinformatics issues as well as the cloud job queuing issues ensured that Musa and I both gained the maximum benefit from the collaboration.

The wide scale analyses were run on the Windows Azure cloud. I modified scripts written in R by Shanahan and Memon to run the analyses. I wrote the C# code necessary to launch web roles on the cloud, as described in section 5.3.2. I wrote scripts to upload data

files from public databases to the Azure cloud storage, ran these scripts to upload data from around 3,000 experiments and ran the various R scripts on the cloud.

Chapter 2

Basic Principles of Molecular Biology, Bioinformatics, Big data, Grids and Clouds

2.1 Introduction

In this chapter an explanation is given of the current basic understanding of the building blocks of life. Starting with the cell as an important functional unit of life, an explanation of DNA (or DeoxyriboNucleic Acid) and some of the mechanisms used in the replication of cells is presented. Section 2.4 will introduce the Central Dogma of Molecular Biology, RNA (RiboNucleic Acid) and amino acids, before explaining what is referred to as a genome. Gene expression is briefly introduced and then an explanation of exons and introns will lead on to an introduction of the field of bioinformatics and some of its tools. The chapter will conclude with a summary of the scale of the big data challenge which faces researchers in bioinformatics today, and outline the basic features of grid computing and cloud computing which have developed to meet the needs of such a challenge.

2.2 The Cell

The number of living species on the earth today is thought to be 5 ± 3 million [5] of which 1.5 million are named. Each species is different, and the members of each one are capable of reproducing themselves. Many living organisms are single cells. Others, such as human beings, comprise many groups of cells which perform specialized functions, and are linked by intricate systems of communication. Whether made of one cell or many million cells, each organism is generated from a single cell originally. This single cell contains hereditary information that defines the species. It also contains the machinery to construct a new cell which is a complete copy of itself.

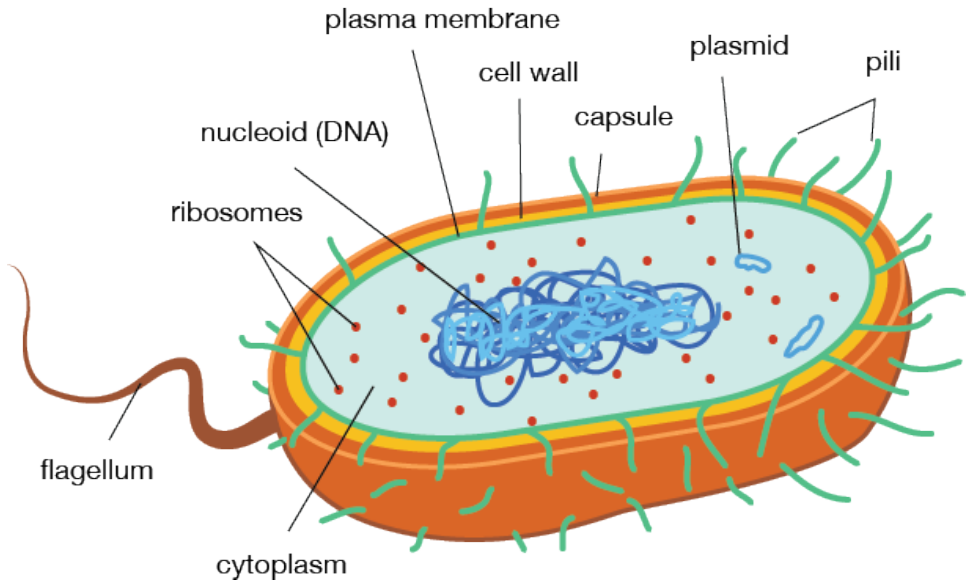


Figure 2.1: Structure of Prokaryote Cells. (Source: Shmoop Biology *http://www.shmoop-.com/biologycells/prokaryoticcells.html*) [Accessed April 8, 2014]

Bacteria, who do not have a nucleus in their cells, are called *prokaryote* cells. Other organisms like mammals, plants and fungi, whose cells are more complex and contain nuclei, are called *eukaryotes*. Figure 2.1 shows some details of typical prokaryote cells. The prokaryote cells of bacteria and archaea which are both microscopic organisms usually shaped like rods or spheres, have only one compartment in the cell. It contains DNA, usually in a single circular chromosome.

Eukaryotic cells, by contrast, are found in many shapes and sizes. They are larger than prokaryotic cells. DNA is stored in the nucleus of eukaryotic cells, in (usually) multiple linear chromosomes with a more complicated gene structure than for prokaryotic cells. Figure 2.2 shows a stylized diagram of typical eukaryotic cells for both plant and animal examples. In both cases the DNA stored within the nucleus is there to give precise information for the replication of cells and for the development of complete living organisms. The work in this thesis concentrates mainly on eukaryotic cells though in Chapter 8 where a wide scale comparison of the data on four different species is described, the bacterium *Pseudomonas aeruginosa* is included as an example prokaryote.

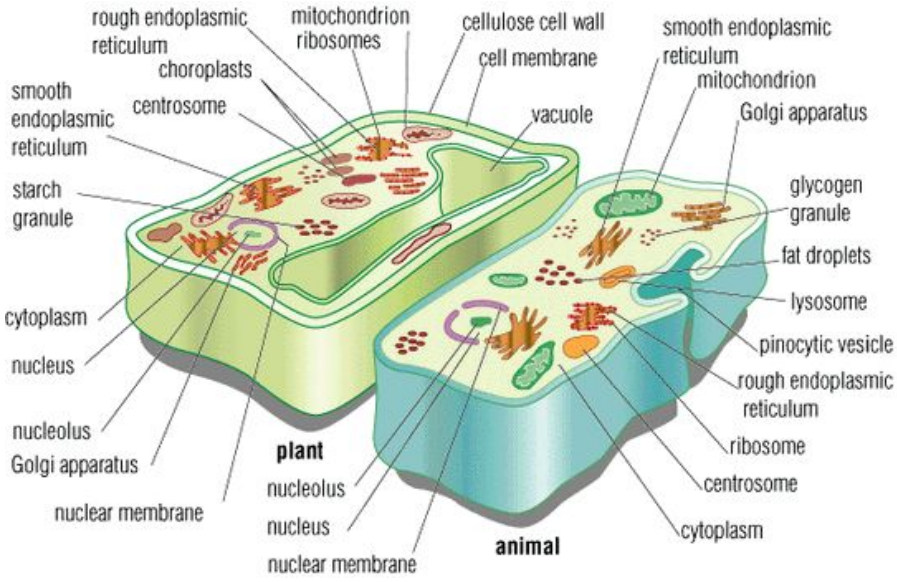


Figure 2.2: Comparison of Eukaryotic Animal and Plant Cells (Source: The Pondering Gulch *http://www.theponderinggulch.com/2011/07/frontyard-senseback-yard-science-getting.html*) [Accessed April 29, 2014]

2.3 DNA

The building blocks of DNA and RNA are shown in Figure 2.3, with details of the bonding of the bases of DNA shown in Figure 2.4. DNA is a double-stranded molecule where each strand consists of a sequence of **nucleotides**. Each nucleotide consists of three parts: a

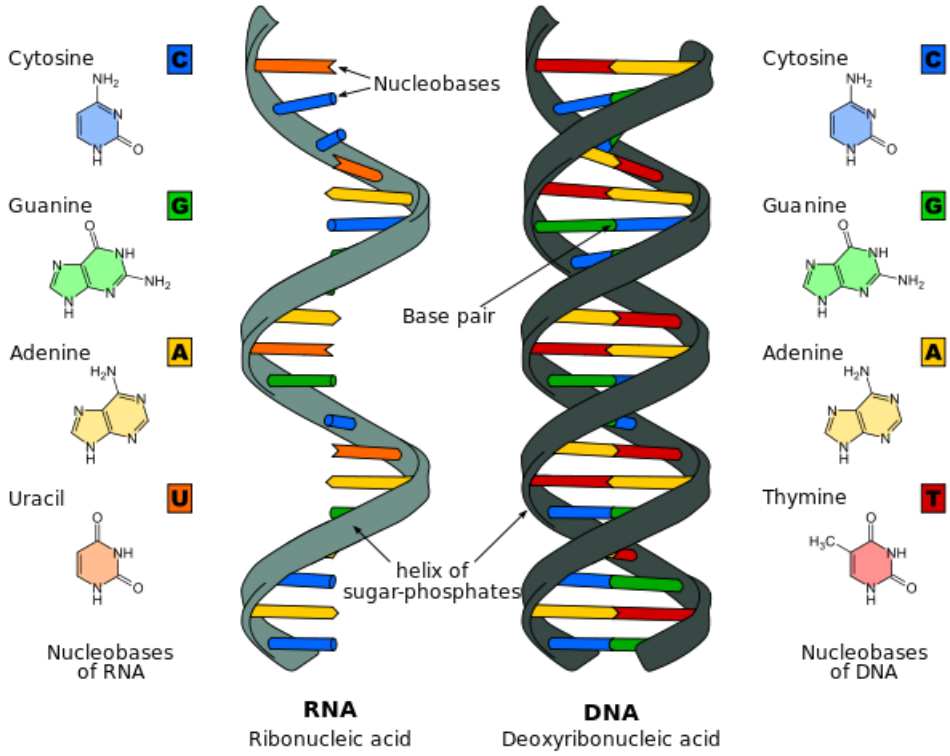


Figure 2.3: Similarities and Differences between DNA and RNA. (Source: Wikimedia Commons) [Accessed April 29, 2014]

sugar (deoxyribose) part attached to a **phosphate** part, and a **base** of either adenine (A), cytosine (C), guanine (G) or thymine (T), so that an example sequence might be GAATTC... The pairing rule of DNA is that A pairs with T and C pairs with G. So in double-stranded form the six base pairs in the example are:-

G A A T T C
| | | | | |
C T T A A G

The nucleobases in Figure 2.3 are referred to as **nitrogenous bases** as they are rich in nitrogen atoms. It can be seen in Figure 2.4 that they hold together across the two strands with hydrogen bonds. The figure shows that adenine and thymine form two hydrogen bonds between them which is represented as $A = T$ or $T = A$. Guanine and cytosine form three hydrogen bonds between them, written as $G \equiv C$ or $C \equiv G$. These two base pairings between A & T and between G & C are known as Watson-Crick base pairs and are crucial

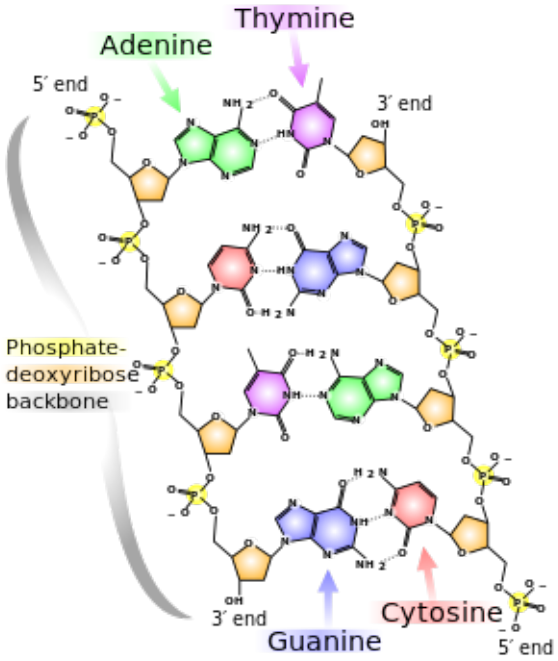


Figure 2.4: Chemical structure of DNA. Hydrogen bonds are shown by dotted lines. (Source: Wikipedia *http://en.wikipedia.org/wiki/DNA*) [Accessed April 29, 2014]

for the DNA double helix structure formation, pictured in Figure 2.3.

In Figure 2.4 the groups of four oxygen (O) atoms with phosphate (P) in the centre are phosphate groups and the pentagons with four carbon (C) atoms and an oxygen atom are deoxyribose (sugar). The sugar part of each DNA molecule binds to the next phosphate part with a covalent bond. This strong sugar-phosphate linkage has led to these two strand parts of DNA being known as the 'backbone'. It should be noted that the sugar-phosphate backbone always gives a direction or polarity to the strand, and that the two ends of a single strand are known as the "$3'$ end" and the "$5'$ end". $3'$ (3 prime) represents the carbon atom in the sugar to which the next phosphate or $5'$ (5 prime) end of the adjacent base attaches. The information in DNA is read and/or copied through the direction from the $5'$ end to the $3'$ end in ways that will be described in the next section.

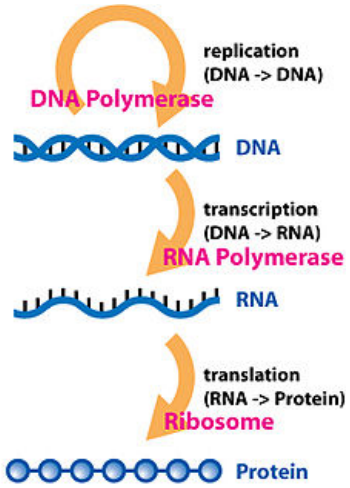


Figure 2.5: The Central Dogma of Molecular Biology with Enzymes. (Source: Wiki-pedia) [Accessed April 29, 2014]

2.4 Central Dogma of Molecular Biology

The **central dogma of molecular biology** is the principle that the flow of genetic information in cells is from DNA to RNA to protein, see Figure 2.5. There are exceptions to this principle in retroviruses for example, whose growth cycle includes a step of copying RNA into DNA by a virus-coded **polymerase**. A polymerase is an enzyme which synthesizes polymers of nucleic acids. **Replication** is the means by which cells multiply themselves through dividing and creating fresh copies which match the original cell in every detail, including the many millions of bases of the DNA strands in the nucleus. DNA polymerase starts and controls the replication process.

Protein synthesis is the term given to the process of **transcription** and **translation** whereby the information in the DNA in cells is used to make proteins as organisms require them. **Transcription** is the method by which cells copy information from DNA into RNA. In eukaryotes it takes place in the nucleus of each cell. Segments of one strand of the DNA sequence form templates so that free nucleotides of RNA can join together in matched pairings with the DNA nucleotides. It is RNA polymerase (RNAP) which initiates the transcription process. The sequence of messenger RNA (mRNA) that is formed then peels away from the DNA and moves out of the nucleus into the cytoplasm of the cell. In RNA

the backbone is formed of a slightly different sugar from that of DNA. It is ribose instead of deoxyribose, and one of the four bases is slightly different: uracil (U) instead of thymine (T). The other three bases, A, C and G, are the same and all four bases pair with their complementary counterparts in DNA. This means that the U of RNA pairs up with the A of DNA, and that the A of RNA pairs up with the T of DNA.

Translation takes place on the **ribosomes** in the cytoplasm. It is a more complicated process than transcription, but similar in that a template sequence of, in this case mRNA, nucleotides is matched with triplet bases of transfer RNA (tRNA) which are available in the cytoplasm to make protein.

The bases of RNA group together in sets of three, known as **codons**. Each codon forms, or **codes for**, a particular amino acid, of which examples are AUA, coding for tyrosine, and UUC, coding for lysine. There are 64, i.e. 4^3, possible codons that can be formed from the four bases, A, U, C and G, but only 20 different amino acids as there are many cases where several codons lead to the same amino acid. Translation causes each codon of the template sequence of mRNA to add a particular amino acid to a growing peptide chain when it pairs up with a codon of tRNA in the cytoplasm. For protein synthesis to be completed, the amino acids have to be linked together as polypeptides and ultimately form proteins in a chain. This completes the translation process. The **ribosome** is like a machine which works along the mRNA template and stitches together the amino acids to form the proteins in their chain.

2.5 Genes and the Genome

A **gene** is a functional unit of the **genome**. For any organism, its genome is its entire DNA. There can be minor variations in the base sequences of DNA between individuals in a species, and these give rise to particular characteristics which may vary between the individuals, for example the colour of eyes or hair in humans. There is much research underway in the field of genomics, where characteristics and the genes which may give rise

to them can be studied. There is also much being discovered in the areas of DNA which do not code for or produce proteins, but still form RNA through transcription and have other functions. The quantity of the regulatory and other noncoding DNA varies widely between different classes of organisms. A gene can also be thought of as a sequence of DNA that occurs in a certain location on a chromosome and determines or partially influences (together with other genes) a particular characteristic of an organism. In all cells, it is the case that there is regulation of protein synthesis. A cell is not producing all possible proteins all the time. Individual genes are **expressed**, i.e. used to make proteins or other gene products, in what could be termed the macromolecular machinery for life, as all life forms use gene expression. The rate of transcription and translation of various genes is adjusted independently by cells as needed.

2.6 Exons and Introns

The sequences that comprise the DNA of any organism do not all code for protein. In eukaryotes the portions of DNA that code for protein are called **exons**, and intervening portions of DNA that do not code for protein are called **introns**, see Figure 2.6.

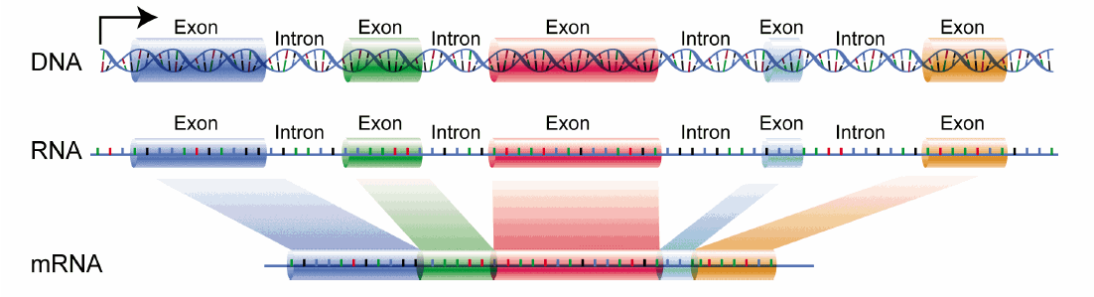


Figure 2.6: Exons and Introns (Source: The National Human Genome Research Institute http://www.genome.gov/Images/EdKit/bio2i_large.gif) [Accessed November 6th, 2013]

In eukaryotic cells, during the process of translation in the nucleus, mRNA is formed from the exonic sequences of DNA only. The introns, so-called because they inhabit the intragenic (inside gene) regions, can be located in a wide range of genes, including those

that generate proteins, ribosomal RNA (rRNA) and tRNA.

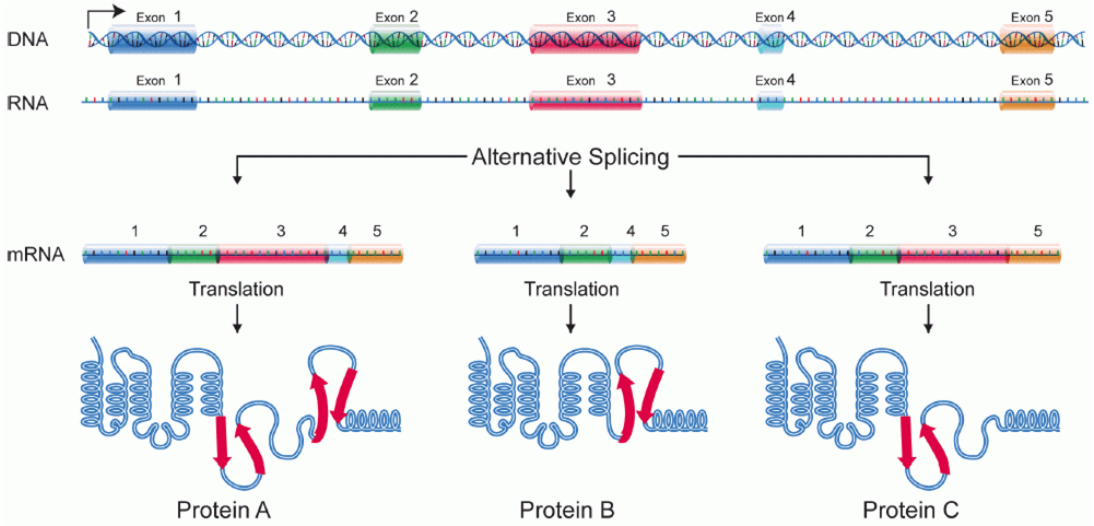


Figure 2.7: Alternative Splicing (Source: The National Human Genome Research Institute http://www.genome.gov/Images/EdKit/bio2j_large.gif) [Accessed June 23, 2014]

During gene expression there is a regulated process called **alternative splicing** by which different proteins may be formed from the same sequence of DNA which makes up part of the DNA of the gene. Figure 2.7 shows an example of alternative splicing where three different proteins may result from the mRNA being transcribed in three different ways from the five exons in the original DNA. This allows more variety in the synthesis of proteins than might be expected.

Polyadenylation is the name given to the addition of multiple adenine (A) bases (called a poly(A) tail) to the 3' end of the RNA formed during transcription. In eukaryotes, polyadenylation is part of the transcription process that forms mature mRNA for translation. As the transcription of a gene finishes, polyadenylation begins. The set of proteins which synthesize the poly(A) tail may add this tail at any one of several possible sites. Therefore polyadenylation can produce more than one transcript for a single gene, which is called **alternative polyadenylation**.

2.7 The “Omics” revolution

“Omics” refers to fields of biological study such as genomics, proteomics and transcriptomics. It has come to refer generally to the study of large, comprehensive biological datasets, though it can infer the analysis of data from more than one of these fields in aggregate. This approach was a conceptual change from biologists conducting experiments on a few tissue samples in a wet lab, to systems biologists running computer models to explain the data results from thousands of microarray or sequencing experiments stored in public databases. Systems biology researchers are able to make use of many types of cellular components of a model organism in this way. The regulation of gene expression, for example, has been studied in many types of cancer cells, but such is the complexity of biological systems that more data and more studies are required to confirm results and further the understanding of cell processes. Needham *et al.* [6] have demonstrated that it is possible to bring multiple studies together to identify some of the subtle changes in gene expression that are biologically meaningful. In this way the so called curse of dimensionality, where output from all the genes is measured but only in a small number of conditions, can be circumvented.

The “omics” sciences include measurements in genomics for DNA variants, in transcriptomics for mRNA, in proteomics for proteins and in metabolomics for intermediate products of metabolism. The technological breakthroughs that allow simultaneous examination of thousands of genes, transcripts and proteins etc., bring many challenges to the understanding of systems as a whole. Major expectations of understanding health issues with attendant improvements in medicine and health have not always been realised. Discoveries have pointed to a highly individualized profile of health and disease, where each case is different, but this has proved difficult to translate into improved personalized healthcare. Sometimes results have been difficult to reproduce in other laboratories, and thus the validity of research results has been questioned.

There are a number of platforms available to study large amounts of biological data. Two of the popular ones for example are Galaxy [7] and Taverna [8]. Galaxy employs a web

portal to give users the opportunity to search remote resources such as genome annotation databases, combine data from independent queries, and visualize the results [7]. Taverna enables bioinformaticians to build workflows or pipelines of services which provide a range of different analyses. These can include sequence analysis and genome annotation [8]. There is even a combined workflow system called Tavaxy [9] which offers new features while integrating existing Taverna and Galaxy workflows in a single environment.

Even when some results have been obtained from the analysis of large amounts of omics data, it can be important that they are confirmed by biologists' repeated studies using hypotheses and confirming them.

The work in this thesis concentrates on transcriptomic data in the form of microarray datasets. This type of data has been widely used in the fields of biology and earth sciences. Its generation and usage will be explained in the next chapter.

2.8 Bioinformatics and some of its tools

Bioinformatics is the research into and application of computational techniques to the data produced by biological investigations. It encompasses a wide range of subject areas from structural biology and genomics to gene expression studies, using statistical and computing methods. Another way to describe bioinformatics is as a management information system for molecular biology, a phrase used by Luscombe in a review paper on bioinformatics [10].

This work concentrates particularly on the informatics of microarrays which will be introduced in more detail in the next chapter. The research used three particular tools which are in common use in bioinformatics and they will be outlined: *BLAST*, R and Bioconductor. As the study began, next generation sequencing was beginning to be more widely employed for many types of genetic research, but there were drawbacks to choosing to analyse this type of data. One was which technology from the different company offerings to choose. Another was that errors could arise in the sequencing process both from the bioinformatic analysis

and from experimental steps [11], and it was early days in the detection and correction of such errors. It was decided to focus on microarray data because the technology had already been tried and tested over about ten years, and some experience had been gained locally into types of bias to which microarray data could be subject [12, 3].

2.8.1 *BLAST*

BLAST is a computer program for searching and comparing base sequences of DNA or RNA. It is freely available to all, both via web interfaces and as an executable program which can be downloaded for use on a local computer. One can use *BLAST* to search and compare a query sequence with known sequences on many reference databases. In this way one can compare the query sequence to find out if it matches or partly matches known genes.

BLAST gives results in various ways. There is a colour-coded score of the number of bases which matched in each database searched, where the higher the score and the colour closest to the colour red indicate the best matches. Matches are recorded by length and by parameters of partial matches. These parameters can help to identify those matches which will be most useful in the current research. One can choose to match sequences to human genomic data, to human transcript data or both. One can also select to search by other organisms.

Megablast is the algorithm which uses larger query nucleotide sequences of DNA to match to reference genome databases. The *megablast* task is optimized for intraspecies comparison as it uses a large word size, whereas *blast* is more suitable for interspecies searches with comparisons which use a shorter word size. *Megablast* was used in the series of procedures to discover the “unique mappings” described in section 3.4.2.

2.8.2 R

R is a computer programming language for accomplishing tasks in statistics, data analysis and graphical display. It can handle data in a variety of forms, from integer and floating

point to arrays and matrices. The R software is free and runs on all common operating systems. It can be used as an interactive command line system, where the user types one line of instructions at a time and waits for the resultant output, or it can be used by submitting a program or pre-prepared sequence of commands which are all executed and output received as directed. **RStudio** is an implementation of R which allows multiple windows simultaneously showing for example an **R script**, a current **interactive window**, **output** such as plots, **help** information about R commands, and **history** information relating to previous R commands used. Thus RStudio is very useful for developing R scripts, for running them on different data and for visualization of the data and computations being studied. In this work R has been used extensively to analyse microarrays and to visualize the collected results.

2.8.3 Bioconductor

Bioconductor is an open source and open development software project for the analysis of genome data (for example sequence, microarray, annotation and other data types). Many packages have been developed and shared from the Bioconductor website, by a large number of different researchers. *affy, affycomp*, and *affyPLM* are examples of Bioconductor packages available for use with Affymetrix microarray data. *affy* has been used in this work.

2.9 The Big Data challenge

In recent years the term **big data** has come to mean any data produced in such large quantities that it poses a significant challenge to process it and extract meaningful conclusions for action. Sometimes big data is spoken of as having a number of **V** properties, for example:-

- Volume: the vast quantity of new data being generated and stored each day.
- Velocity: the increasing speed at which data is generated and at which it can be processed.

- Variety: data can be structured or unstructured and can vary from text to geo-spatial form, or from tweets to photos and videos.
- Variability: the meaning of some data can vary depending who collects it. Data can also be variable depending on when it was collected in the same sense that language can change the meaning of words over time.
- Veracity: data is only useful if it is accurate. Often data is found to be messy because of errors and inconsistencies within it.
- Visualization: after data has been processed, it needs to be presented in a readable and accessible way. New visualization packages are being developed every year.
- Value: data is only as valuable as the accurate insights and information that it provides. Big data can be hugely valuable, but only with the analysis tools that unlock its information.

The European Bioinformatics Institute (EBI) in Hinxton, UK, stores over 20 petabytes (1 petabyte is 10^{15} bytes) of data and backups about genes, proteins and small molecules, according to an article by Vivien Marx in the June 2013 edition of *Nature* [13]. The data are used heavily by scientists around the world who are working in both academia and in industry. The nucleotide sequence databases "have a doubling time of less than one year" according to the EMBL-EBI annual report of 2012-2013 [14].

Breakthroughs are being made in many areas because of the large amount of research data that is publicly available at data centres such as the EBI. In the area of computational biology and bioinformatics there have been further major successes in the ENCODE project (ENCyclopaedia Of DNA Elements) which was planned as a follow-up to the Human Genome Project. One example is the production of a detailed map of human genome function [15]. A major analysis of the gut metagenome was performed by the Structural and Computational Biology Unit at EMBL Heidelberg, so that more than ten million mutations in the bacterial strains in the gut of 207 individuals were identified [16]. Another group

devised a method to store information in synthetic DNA, which might provide the technology for long-term storage of infrequently accessed or archive data [17]. The BGI (formerly the Beijing Genomics Institute) in Shenzen, China, is the world's largest genetic research centre. It generates at least a quarter of the world's genomic data, having 178 machines [18] (in January, 2014) to sequence the genomes of samples from people, plants, animals and microbes.

With so much data being generated on the planet, there is much scope for computational biologists to make discoveries using other people's data. Much data sits "under-analysed in databases all over the world" says Marcie McClure, a computational biologist at Montana State University in Bozeman [13]. McClure and her team have discovered eleven new fish retroviruses by analysing genomes computationally. Some of the approaches and tools of bioinformatics will be outlined in the next section.

The challenge big data presents is how best to efficiently analyse it in the different fields of research that it can benefit. Most researchers have historically tended to download data to their local machines for analysis. But this method is "backward" according to Andreas Sundquist, chief technology office of DNAnexus [13], because "the data are so much larger than the tools, it makes no sense to be doing that." The alternative is to use a grid or a cloud for both data storage and for the analysis of the data. Hopefully the time and cost of accessing large amounts of data for computation will be reduced when the data resides 'near' the compute facility. The proximity of data to the CPUs of clouds will depend on how the cloud provider has organized their data centres and their back-up resources. Cloud prices for data storage, data access and computation tend to reflect a provider's methodology or business model. More about grid computing and cloud computing will be discussed in chapters 4 and 5. A brief introduction to the concepts of grid and cloud computing will be given in section 2.10.

2.10 The computational challenge: grid computing

In 2009, near the beginning of this doctoral research programme, cloud computing was in its infancy. There were some networks available to assist researchers who did not have access to enough computing power in their own institution for their requirements. These loosely connected computer networks known as **grids** had been available for a few years and were being found helpful for sharing computing resources across academic communities. The development of software on grids for accessing computing resources across the internet was ongoing, and there were several different methods in use for authorising access to remote computers and maintaining security.

Clouds and Grids have been compared and contrasted in many academic studies in recent years (for example: by Foster *et al.* [19] and by Kondo *et al.* [20]). Both can offer a service to users who need more computing resources than they have to hand in their local situation. This section will describe the features of grids.

2.10.1 Definition of a Grid

There are various different definitions of grids which have been proposed. One important checklist for evaluating grids is given by Foster [21]. This says that a grid is a system that:-

1. **coordinates resources which are not subject to centralized control**

   This allows resources which may be spread geographically and owned by different parties to be shared within a grid arrangement. A user can link from their own desktop to resources which are not necessarily owned or managed by their own enterprise. The grid addresses the issues of security, payment, policy and membership that arise in these settings.

2. **uses standard, open, general-purpose protocols and interfaces**

   The fundamental issues of authentication and authorization are handled by a grid using interfaces and multi-purpose protocols. It is important that these protocols

and interfaces be standard and open for the wider development of grid access. A grid must also address resource access and resource discovery to make its facilities directly available to users. It is standards that allow grids to establish resource-sharing arrangements dynamically with any interested party. They are also important to enable general-purpose services and tools.

3. **delivers non trivial qualities of service**

   A grid should allow its constituent resources to be used in a coordinated fashion to deliver various qualities of service. These can relate to response time, throughput, security and availability of services. The complex demands of users may need to be met via the allocation of multiple resource types. Foster [21] believes that the utility of the combined grid system should be significantly greater than that of the sum of its constituent parts. Fault tolerance and stability are other issues to be addressed within the qualities of service provided.

The Open Grid Forum *(http://www.ogf.org/)* was developed to enable progress on standards for grids, and several years of experience and refinement produced the widely used standard, the open source Globus Toolkit *(http://toolkit.globus.org/toolkit/)*. Work continues on standards for grid computing, both in IT companies and in academic research groups.

2.10.2 Survey of Grids

2.10.2.1 Grid types

It is useful to describe grid systems in terms of the functionality which they provide, as shown in Figure 2.8, a taxonomy proposed by Krauter *et al.* [22]. The **computational Grid** category refers to systems which have higher aggregate computational capacity available for single applications than the capacity of any constituent machine in the system. It is possible to subdivide these systems further into **distributed supercomputing** and **high throughput**

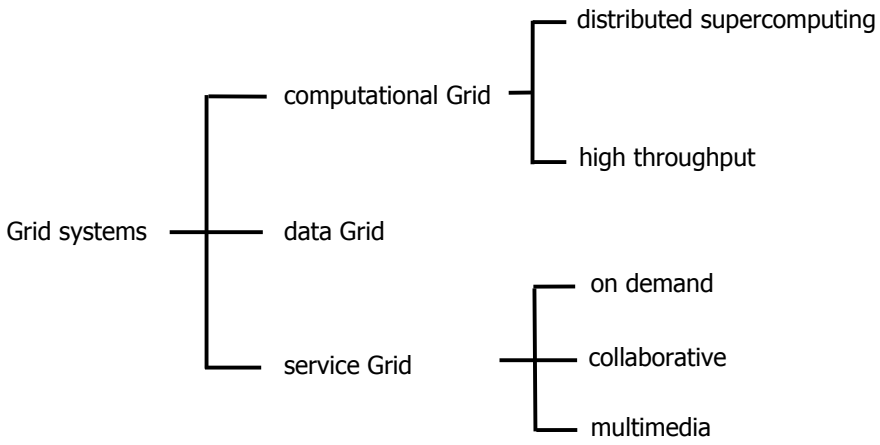


Figure 2.8: A Grid systems taxonomy

systems. A distributed supercomputing grid has the capacity to execute the application in parallel on several machines in order to reduce the completion time of a job. Typically, the large scale simulation problems such as weather forecasting need this type of grid system. The high throughput grid category is able to increase the completion rate of a stream of jobs.

The **dataGrid** category provides the specialized infrastructure to applications for storage management and data access. There are dataGrid initiatives such as the European DataGrid Project [23] and Globus [24] which work on developing large-scale data organization, management, catalogue and access technologies.

The **service Grid** category is for systems which offer services not provided by any single machine. An **on-demand** grid category is able to dynamically aggregate different resources to provide new services, for example when a researcher wants to allocate more machines to a simulation. A **collaborative** grid is able to connect users and applications into collaborative workgroups. These grids enable real time interaction between humans and

applications via a virtual workspace. A **multimedia** grid is able to supply an infrastructure for real time multimedia applications. This requires a quality of service to be supported across several different machines.

2.10.2.2 Grid architecture

A grid's architecture is sometimes described in terms of "layers", where each layer has a specific function. The higher layers are those which interact with users, whereas lower layers are those which manage the computers, data and networks.

- The lowest layer of grid architecture is known as **the network** which connects grid resources.
- The **resource layer** lies above the network layer and consists of the actual grid resources such as computers, storage systems, data catalogues, sensors and other instruments that might be connected to the network.
- The **middleware layer** provides the software and hardware tools that enable the various elements of the grid such as servers, storage and network components, to participate in a grid. It is a vital control and management layer.
- The highest layer of the structure is the **application** layer, which includes portals and development toolkits to support applications and development as well as the applications themselves. Users interact with this layer, which can also provide information about the usage of elements of the grid, speed of service and other tracking data.

2.10.2.3 Grid resource and scheduling organization

The methods of organising resources and of scheduling jobs are discussed in detail by Krauter *et al.* [22]. Resources can be managed either by a schema based approach or by an object model. In a schema based approach, the data that makes up the resource is described

by a description language together with some integrity constraints. In an object model, the operations on the resources are defined as integral to the resource model. The object model can be predetermined and fixed as part of the definition of the Resource Management System (RMS). Both the schema and the object model approach can be extensible so that new schema types or new definitions can be added.

The scheduler organization can be centralized, hierarchical or decentralized. In the centralized organization, there is only one scheduling controller which takes responsibility for the decision making system-wide. An organization like this has several advantages which include easy management, simple deployment and the ability to co-allocate resources. Some disadvantages include the lack of scalability, lack of fault-tolerance and the difficulty in accommodating multiple policies. The other two organizations, hierarchical and decentralized, have more suitable properties for a grid RMS scheduler organization. In a hierarchical organization the controllers are designated to manage defined sets of resources which addresses the issues of scalability and fault-tolerance. The decentralized organization is able to address issues such as fault-tolerance, scalability and multi-policy scheduling, but introduces some problems of its own such as management, usage tracking and co-allocation. Protocols are required to manage the scheduling on large network sizes, and the overhead of operation of these protocols is a determining factor for the scalability of the overall system.

The rescheduling characteristic of a RMS determines when the current schedule is re-examined and jobs reordered. Job executions can be reordered in order to maximize resource utilization, job throughput or other metrics depending on the scheduling policy. The interval between rescheduling can be either periodic or event-driven depending on which approach delivers the quality of service required.

2.10.2.4 Secure access

Secure access to shared resources is one of the most challenging areas of grid development. In order to gain secure access, grid developers and users need to be able to manage three

important things:-

1. Access policy: What is shared? Who is allowed to share? When can sharing occur?
2. Authentication: How does one identify a user or resource?
3. Authorization: How does one determine whether a certain operation is consistent with the rules?

Grids need to save and track all this information, which may change from day to day. Therefore grids need to be flexible and have a reliable accounting mechanism. It may be that pricing policies will be decided by using this information.

These accounting challenges are not new, but in the context of shared resources they present a complex picture. The issue of security is linked to trust. One may trust the other users, but can one trust that one's data and applications are securely protected on their shared machines? New security solutions are constantly being developed, including sophisticated data encrytion techniques, but it is a never-ending race to stay ahead of malicious hackers.

2.11 The computational challenge: Cloud computing

There is no single widely accepted definition of Cloud computing. However the National Institute of Standards and Technology in the USA has defined it like this: "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [25]

Cloud computing is a way of delivering computing resources to an end user without that user, or their organisation, having to invest large capital resources into their own hardware and software. The term, **Computing as a Service**, or CaaS, serves to describe the general overall service that cloud computing provides. It is typically accessed over a network such

as the internet, with the computer processors and storage at a location which may be distant from the users.

While grid computing can be used to harness the power of multiple computers in many locations to all work on the same compute-intensive job, cloud computing tends to be the solution for small and medium sized enterprises to scale up and scale down their application requirements as needs vary. Clouds can be built on a grid system, but not vice versa.

Data Centres have been providing computer processing resources to their clients for many years, with the advantages of storage backups, security and the cooling of machine rooms being handled by the Data Centre rather than by the client. Larger companies have run their own Information Technology (IT) departments to supply computing facilities around their various company premises. However, during the rapid expansion of micro computers in the 1980s, most companies began distributing their computing facilities around departments and offices so that all users had computer processors rather than just a terminal connected to a central main-frame computer.

2.11.1 Types of clouds

It is possible now to see at least four types of computer cloud in use: public, private, community and hybrid.

- A **public** cloud is one which offers computing services as a commercial business to the general public or to an industry group.
- A **private** cloud is one which exists for its own user base within an organisation. It may be managed by the organization or by a third party, and it may reside on or off the premises.
- A **community** cloud is one whose infrastructure is shared by several organizations. It supports a specific community which has shared concerns such as policy, mission,

security requirements or compliance considerations. It may be managed by its community organizations or a third party, and may exist on or off premises [26].

- A **hybrid** cloud is a combination of two or more or the above types of cloud that remain unique entities but are bound together by standards that enable data and application portability. An example of a hybrid cloud is an organisation which runs its IT services as a private cloud, and at times of high activity can automatically buy services from a public cloud to meet service demands of users, rather than build up longer queues or extensive response times.

In recent years the cloud computing concept has gained ground with several large companies offering worldwide public cloud services which provide advanced Data Centre type computing facilities to their clients. In this chapter some characteristics of clouds will be explained and two public cloud computing offerings will be introduced: Amazon Web Services and Microsoft's Windows Azure. A local private cloud will be described that was built by Ibrahim Musa, a PhD student in the Computing Science and Electronic Engineering (CSEE) department of the University of Essex, with whom a collaboration was arranged. This collaboration enabled some cloud research using the human microarray data that will be analysed again in other chapters.

2.11.2 Characteristics of cloud computing

Cloud computing typically exhibits some of these, mostly beneficial, characteristics:-

- **Flexibility**

  One can choose the operating system and language that makes the most sense for any application. There is also a choice of services to use for computing and for storage, allowing the flexibility to concentrate on innovation, not infrastructure.

- **Cost reductions**

Applications can be deployed quickly and easily, so if workload has peaks at certain times of the day, one can cover these peaks yet deploy fewer machine instances and save costs at other times.

- **Reliability**

  Reliability is important to all customers, so cloud providers try to build in redundancy to ensure that their infrastructure is always available for customer needs. For example, the Amazon EC2 Service Level Agreement commitment is 99.95% availability for each Amazon EC2 Region, according to their website *http://aws.amazon.com/-ec2/* on 17th March, 2014.

- **Scalability**

  This feature of cloud computing is attractive to customers whose computing needs fluctuate through each 24 hour period. Applications can be quickly deployed, scaled up or scaled down as demand dictates. One may need one virtual server or thousands at any time, and this fluctuation in demand can be handled.

- **Performance**

  Different performance levels can be gained depending on the type of application being run, and the size of machine (instance) being deployed. An instance can vary from 'Small', with the following features, for example (examples are taken from the Microsoft Windows Azure website *http://msdn.microsoft.com/en-us/library/windows-azure/dn197896.aspx* on August 15th, 2013):-

  - 1.7 GB memory
  - 1 Compute Unit (1 virtual core with 1 compute unit)
  - 160 GB instance storage
  - 32-bit or 64-bit platform
  - I/O Performance: Moderate

to ‘Extra Large’, with these features:-

- 15 GB memory
- 8 Compute Units (4 virtual cores with 2 compute units each)
- 1,690 GB instance storage
- 64-bit platform
- I/O Performance: High

There are other variations on these standard instances such as **Micro instances** which are useful for lower throughput applications, and **High Memory instances** for high throughput applications, such as database and memory caching. **High-CPU instances** have more CPU resources than memory (RAM) and tend to be well suited for number crunching programs. **Cluster compute instances** provide high CPU resources together with increased network performance and are suited to HPC (High Performance Compute) applications. It is vital to choose carefully the type and number of instances to deploy, and to monitor their use, in order to achieve good performance.

- **Easier maintenance**

  Cloud computing reduces the need for significant hardware maintenance on the user’s site. There can also be savings on software maintenance, depending on the level of service being used on the cloud (see sections 2.11.3.1 to 2.11.3.3).

- **Device independence**

  Cloud computing resources can be accessed from any type of computer on the internet. Provided that the computer has a web browser and an internet connection, it does not matter if it is a traditional desktop or laptop PC, a netbook, tablet, smartphone, e-book reader or any other sort of cloud access device.

- **Location independence**

  The device independency means that a user can access the same cloud resources from their home, work or travel locations, knowing that they will always have access to the latest versions of their files.

- **Security of data**

  A cloud provider is paid for reliability as mentioned above, so the headaches of data security are mostly removed from the user. This can mean that one does not have to plan to make backups of vital business data, or consider backup storage in more than one location in case of fire. These can be addressed in the level of service contract that is agreed with a cloud service provider.

- **Security of access**

  Data is often confidential to an organisation, whether for business reasons or for personal privacy reasons. Some organisations have been slow to move to cloud solutions for their data because of a lack of trust in the cloud provider to be able to guarantee secure storage and secure data transfers. This problem is continually under debate across the internet, with spammers and hackers sometimes grabbing headlines in the news.

- **Service loss or interruption**

  There can be slow downs or interruptions to service on the internet, or even a loss of service such as happened to some customers of the Amazon cloud in April 2011 *(http://money.cnn.com/2011/04/29/technology/amazon_apology/)*. Hopefully these events will be rare, but they are a factor to be considered when deciding to use cloud solutions.

2.11.3 Organisation of cloud services

There are many ways that cloud computing can be organised and each has its benefits and drawbacks to the users and to the cloud providers. Three commonly offered service models are described here: Infrastructure as a Service (IAAS), Platform as a Service (PAAS) and Software as a Service (SaaS). Figure 2.9 shows these three types of service in columns, with the additional column on the left called "Packaged Software" which represents the situation where the user manages all software themselves without the benefit of any cloud services. The green shading in each column of Figure 2.9 represents the layers of cloud computing which are supplied and organised by the user, or customer. The brown shading in each column represents the layers which are supplied and managed by the cloud provider.

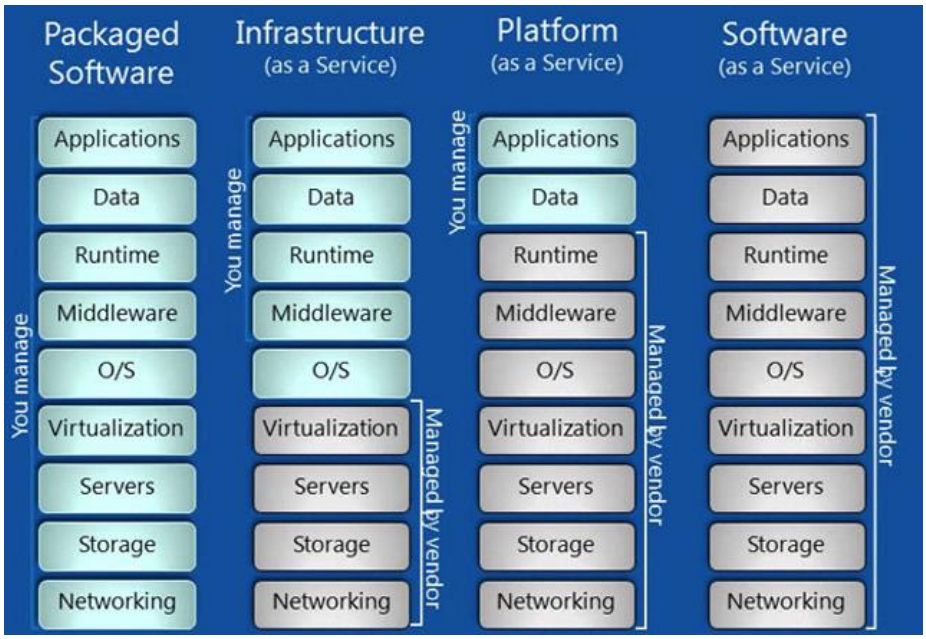


Figure 2.9: Simplified explanations for the three main layers of cloud computing (*http://venturebeat.com/2011/11/14/cloud-iaas-paas-saas/*) [Accessed September 17th, 2014]

2.11.3.1 Infrastructure as a service (IAAS)

IAAS (Infrastructure as a service), as the name suggests, provides the computing infrastructure for performing tasks. This means that physical or (quite often) virtual machines are provided alongside network and other resources like virtual-machine disk image library, block and file-based storage, firewalls, load balancers, IP addresses, and networking. Examples of IAAS services are: Amazon EC2, Windows Azure, and Rackspace. With this type of service, customers typically have to provide their own web interface, operating system and application programs, but they gain the Cloud advantages of scalability so that variations in demand for their applications are automatically handled and they only pay for the resources used.

With IAAS a customer can outsource their hardware needs to a cloud provider who maintains an off-site server, storage and networking hardware. The customer is freed from providing machine rooms for servers, air conditioning and security for these machines. They can run their applications on the cloud hardware and access it at any time over the internet.

2.11.3.2 Platform as a service (PAAS)

PAAS (Platform as a service) provides computing platforms which typically include the operating system, the programming language execution environment, a database and a web server etc. Examples of PAAS are: AWS Elastic Beanstalk, Microsoft Azure, Heroku, AppFog and Google App Engine.

The cloud provider can offer assistance with web application development but the main offerings are a wide variety of solutions for developing and deploying applications over the Internet, such as virtualized servers and operating systems. This saves money on hardware and makes collaboration easier for a scattered workforce. With this level of service the customer just manages their applications and their data as shown in Figure 2.9.

2.11.3.3 Software as a service (SAAS)

Software-as-a-Service is a category of cloud computing which enables users to consume software on a pay-per-use model, while the vendor maintains the software and the hardware. *Salesforce* is an example from the business community. Consumers don't have to concern themselves with either the hardware or the software which is providing the application over the internet to their desk or mobile computer. There are hundreds of thousands of business and personal applications already available at this service level. Personal examples of SAAS are: Netflix, MOG mobile music app, Google Apps, and Dropbox.

2.11.4 Choice of cloud services

For this research, the Amazon cloud services were available in an early form, so their use was investigated after a grid computing trial on the NGS (National Grid Service for the U.K. academic research community) which is described in chapter 4. Later the Azure cloud from Microsoft became available, and a collaborative project under the Venus-C initiative between Dr Andrew Harrison and Dr Hugh Shanahan (of Royal Holloway College, London University) enabled the storage of large quantities of data on the Azure cloud for a limited two year period. The project was able to use substantial computing resources on Azure which enabled a wide scale survey of microarray data. This would not have been possible at the time on local computing resources.

Although the type of operating system did not have a particular bearing on the choice of grids or clouds used, it is noted that the NGS grid offered a range of machines running either Linux systems or Windows systems. The Amazon cloud also offered machine instances of either Linux or Windows. The Azure cloud was only offering Windows services at the time it was used. The local private cloud used Linux systems.

The next chapter describes the nature and use of the microarray data chosen for research. Some more detail of grid computing is given in chapter 4. Chapters 5 and 6 focus on clouds.

Chapter 3

Microarray Informatics

3.1 Background to the analysis of microarrays

A search of PubMed using "microarray data" in abstracts shows over 23 thousand papers and articles (precisely 23,319 for a search performed on 23 April, 2014). Searching for "microarray" alone shows over 60,000 articles. As well as providing information about gene expression in different conditions, microarray data has proven enormously useful in terms of functional annotation (the classifying and characterizing of functions of a DNA sequence, for example in terms of protein coding) and gaining a deeper understanding of processes including the cell cycle.

Using these data sets to infer a network of interactions between gene products, has been one of the major challenges of systems biology. In the field of Evolutionary Systems Biology, comparisons of conserved expression patterns across different species are providing important insights. The unraveling of genotype-phenotype relations in the field of genetics, for example, has been undertaken by Volkers *et al.* in analysing large publicly available gene expression datasets to understand aging in *C. elegans* [27].

In the field of medical research, Atul Butte and his team at Stanford University identified a new gene implicated in type 2 diabetes, by looking at correlations across gene expression experiments in the public domain [28]. The authors of each of the original studies had

missed this gene.

Such studies employ transcriptomic data from a wide number of experiments (in contrast to most transcriptomic studies where the emphasis is on one relevant experiment). On the basis of coexpression (the simultaneous expression of two or more genes) over many experiments, relationships are inferred between genes, and particularly between gene products.

Importantly, in the above work, the variable quality of the datasets used has not been examined. It is implicitly assumed that all of the data used gives an accurate (or at least reasonable estimate) of the expression levels of the genes considered. An improvement in the underlying quality of the data will have a positive down-stream effect on all the work carried out in this area, and hence is a complement to the main research carried out in Systems Biology.

3.2 Microarray technology

Microarray technology enables the simultaneous analysis of thousands of genes in a single reaction quickly and efficiently. It was originally a cottage industry in which microarrays or "chips" were made in individual laboratories by researchers who then used them and analyzed the results, according to Gautier *et al.* [29]. Microarrays were then developed commercially by a number of companies, of which Affymetrix is one well established example. Indeed Affymetrix has a registered trademark for the name **GeneChip®** for its microarrays. Figure 3.1 shows a photograph of an Affymetrix GeneChip® at approximately its actual size.

A typical microarray experiment involves the hybridization of a mRNA molecule to the DNA template from which it originated. Up to millions of **probes** consisting of short DNA sequences are typically fixed on a slide prior to hybridization and the results are then used to calculate estimates of gene expression. Microarrays can be fabricated in a number of

Figure 3.1: An Affymetrix GeneChip®(Source: *https://www.flickr.com/photos/jseita/-3764113525*) [Accessed February 20, 2015]

ways, including printing with fine-pointed pins on to glass slides, photolithography using pre-made masks (as practised by Affymetrix), photolithography using dynamic micromirror devices, ink-jet printing or electrochemistry on microelectrode arrays.

In **spotted microarrays**, the probes are oligonucleotides, cDNA or small fragments that correspond to sections of mRNAs. The probes are synthesized before being "spotted" on to glass. Researchers can produce "in-house" printed microarrays from their own labs. In this case the arrays may be easily customized for each experiment. However in practice these spotted microarrays were shown by Bammler *et al.* not to provide the same level of sensitivity compared to commercial oligonucleotide arrays [30].

Microarrays are often referred to as GeneChips "because they are built on technologies adapted from the semiconductor industry - photolithography and solid-phase chemistry" says Jeff Augen [31, p. 195]. Each small quartz wafer array contains densely packed oligonucleotide probes whose sequences are chosen to match specific genes. The Affymetrix GeneChip is very popular among the microarray technologies available.

The analysis of GeneChips is a multi-faceted challenge, requiring insights from genomics, biophysics and statistics. In order to obtain a summarized estimate of each gene, one

has to apply a complicated pipeline of steps that are not necessarily optimised for associative studies. This means that the biological experiments giving rise to the GeneChips are all independent, and researchers have not all necessarily made the same assumptions or followed the same procedures in preparation or processing of the GeneChips (see section 3.2.2 for an explanation of typical preparation steps for an experiment).

Researchers can purchase high quality microarray chips, together with some equipment and proprietary software to perform analysis upon the microarray data. It is important to consider the analytic procedure performed by the software, so that one does not make false assumptions about the statistical results of the analysis, a point underlined by Gautier *et al.* who were involved in developing analysis software such as *affy* mentioned in section 2.8.3 for Affymetrix GeneChip data [29].

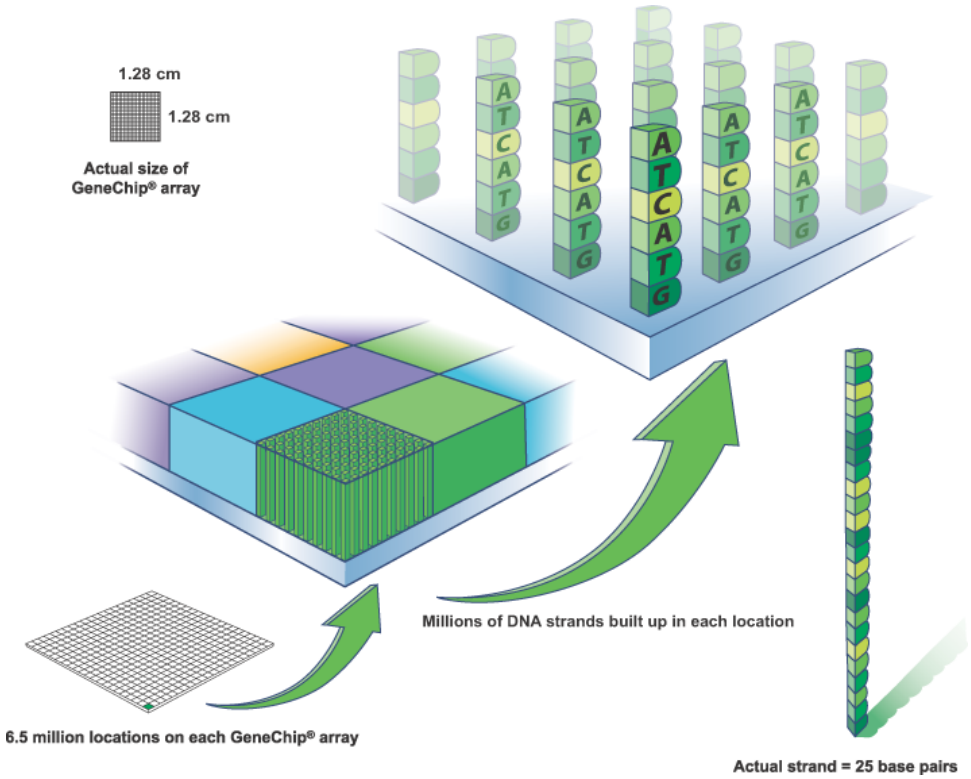


Figure 3.2: The small quartz wafer microarray contains thousands of 25-mer probes in each tiny square of its array (Courtesy of Affymetrix, Inc., Santa Clara, CA, USA *http://media.affymetrix.com/media/corporate/media/imagelibrary/high-res/singlefeature.zip*) [Accessed April 9, 2014]

The work described in this thesis used Affymetrix GeneChips whose nature and pro-

cessing are shown in Figures 3.2 to 3.4. Each microarray is a small square quartz wafer divided into millions of locations. Each location contains millions of DNA strands which Affymetrix has chosen to be a sequence of 25 nucleotide bases. The GeneChips are prepared with the probes being laid down one nucleotide base at a time, using masks supplied for that type of GeneChip by Affymetrix. Figure 3.2 shows an enlargement of the probes on a prepared GeneChip.

Tissue samples of interest to the researcher are produced and labelled with appropriate dyes. Then fragments of RNA are extracted from the tissue samples and hybridized to the microarray in a solution which is incubated for 12 to 24 hours at between 45°Cand 65°C, see Figure 3.3 which shows some probes as RNA fragments are attaching themselves during hybridization. The array is then washed to remove any of the sample which is not hybridized to the probes (known as features).

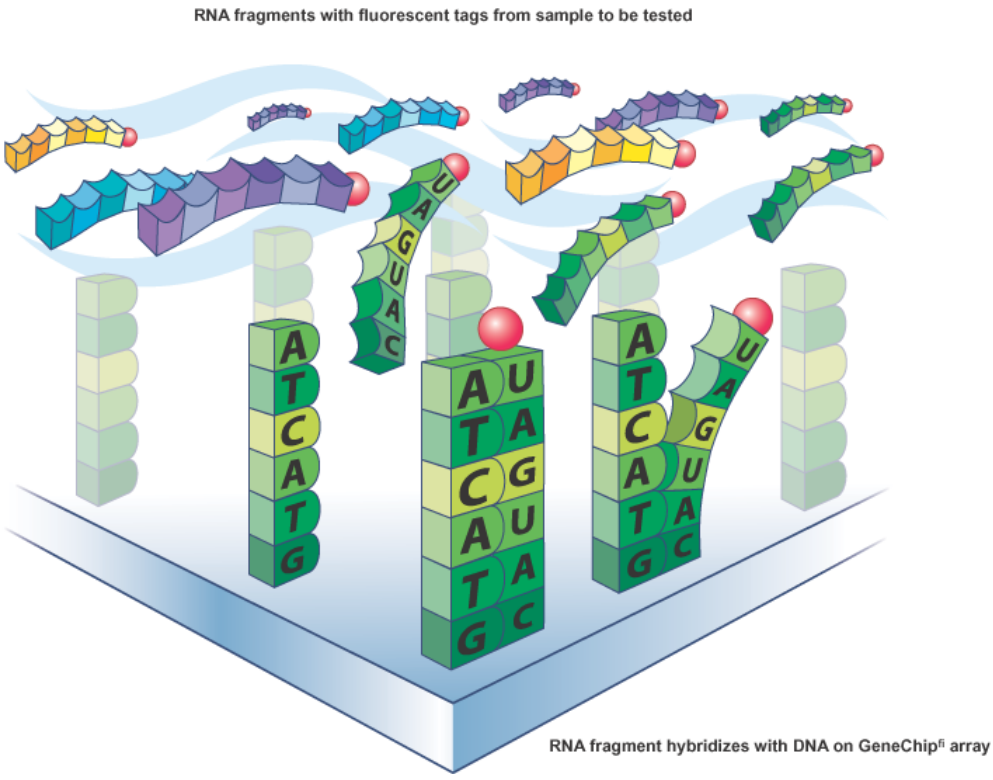


Figure 3.3: Affymetrix GeneChip Hybridization: fragments of RNA stick to the probes (Courtesy of Affymetrix, Inc., Santa Clara, CA, USA *http://media.affymetrix.-com/media/corporate/media/imagelibrary/highres/hybridizationoftag-gedprobes.zip*) [Accessed April 9, 2014]

The final step of the GeneChip processing is to produce an image of the surface of the hybridized array. The array is placed in a scanner which uses a laser light to excite the dye at each pixel (probe) of the array. The amount of fluorescence (see pink tops on some features in Figures 3.3 and 3.4) or light intensity of each probe is detected by a photo-multiplier tube in the scanner and recorded. Software supplied by Affymetrix (such as MAS5, described in section 3.3.2) is used to process the raw data from the image scanner and store the intensity values and other relevant data into standard files called CEL files (described in section 3.2.6, with more detail in Appendix A).

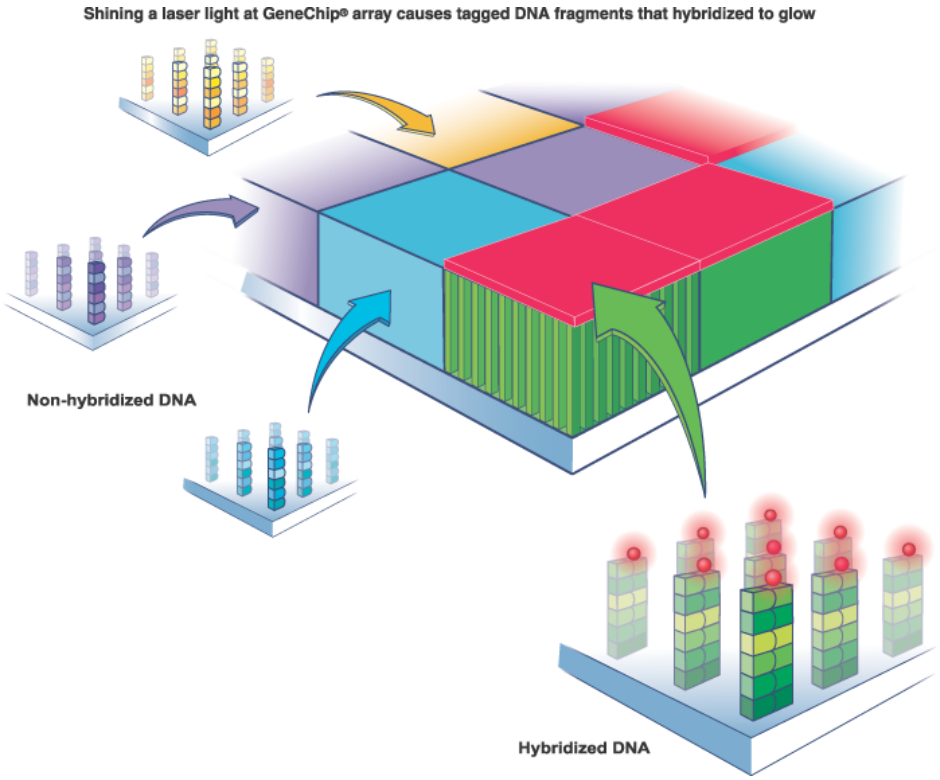


Figure 3.4: Hybridized DNA fragments glow when a laser light is shined on to a microarray, which contains many millions of fragments (Courtesy of Affymetrix, Inc., Santa Clara, CA, USA *http://media.affymetrix.com/media/corporate/-media/imagelibrary/highres/taggedanduntagged.zip*) [Accessed April 9th, 2014]

3.2.1 Elimination of noise in microarrays

The presence of noise can be a problem in large scale microarray studies, and can originate from several different sources. There can be some background fluorescence due to various reasons such as some optical noise from the scanner [32], or some deposits left after washing the chip. Noise due to non-specific binding of mRNA fragments to probes which are not targeting them is a typical source highlighted by Wu *et al.* [33], which was alleviated through the use of probes on the chip designed to counteract this problem. Each such probe, called a Perfect Match (PM) probe, was paired with another probe, called a Mis-Match (MM) probe. MM probes were identical to PM probes except for the middle (13th) nucleotide in the sequence, which was changed for its complement (A for T, or C for G, for example). The value for the true signal was suggested by Affymetrix to be obtained by subtracting the signal for the MM probe from the signal for the PM probe. The value of PM and MM probes was subsequently proved to be less useful than originally planned. Li and Wong [34] showed that PM-MM difference values were highly variable and indeed that the variation due to probe effects can be larger that the variation due to arrays. In addition to noise, there are other reasons why some reported intensity values might be misleading on GeneChips. Upton *et al.* [32] showed for example, that:-

- scanners could lose focus if not frequently maintained and cause errors in intensity readings
- intensity values could be affected by neighbouring probes having high intensities
- some probe sequences can show abnormal binding affinities (and hybridize to neighbouring probes rather than the desired RNA sequences)
- some probes can cross-hybridize with the primer-spacer appended to transcripts.

3.2.2 Planning and conducting an experiment using microarrays

The following considerations are important in planning and conducting an experiment with Affymetrix microarrays:-

1. The experiment must be designed with an end point in mind. What biological conditions are to be tested? Are these conditions continuous or discrete? How many replicates will be necessary (because as the number of conditions increases, the number of experiments required will increase)?

2. The sources of experimental variability should be considered and minimised in accordance with planning the hypotheses to be tested. Technical variability must also be minimized, for example, sample quality control, training of personnel, and calibration of equipment.

3. The biological variability must be considered, e.g. gender, diet, cell cycle patterns and time of day, so that biological replicates are "true" replicates.

4. Having designed the experiment, tissue samples are produced and the Affymetrix GeneChips are prepared and used in accordance with the description above.

5. Data analysis is performed to calculate gene expression, the up-regulation or down-regulation of particular genes and to what extent the hypothesis planned in the experimental design has been supported.

3.2.3 Public repositories of microarray data

Many researchers store the results from their microarray data in public databases. The following are key repositories of Affymetrix microarray data, though they also hold data from other formats and other commercial microarrays. The National Center for Biotechnology Information (NCBI) in Bethesda, Maryland, U.S.A., hosts the Gene Expression Omnibus (GEO) respository [35] which contains microarray data for many GeneChips. This

data is publicly available over the internet. Much of it is also available from the European Bioinformatics Institute (EBI) at Hinxton, Cambridge, U.K. ArrayExpress [36] is the name of the database at the EBI which stores data on functional genomics experiments, including a copy of the Gene Series Experiment (GSE) data from GEO. Each experiment or dataset comprises some metadata about the experiment, the summarized results and usually a set of raw measurements which are stored in CEL files, explained in section 3.2.6.

3.2.4 Design of Chip types

Affymetrix has designed its microarrays so that they are known by their **chip type**. Each chip type is uniquely designed for a particular species genome and a particular purpose. The chip type and its content in terms of the probes, the probe sequences, the probe locations and the probe sets do not change during the lifetime of the array and array type. Examples of chip types are HG_U133A and HG_U133_Plus2 for two different human arrays.

It is important to consider how Affymetrix choose their probe sequences when designing a new GeneChip. The probe sequences are chosen so that the sequences for all probes within a probe set target different parts of the same gene. The suggestion is that many probes targeting a single gene results in many measures whose combination will give a better estimate of true gene expression than any single measure. Early chips were improved upon by the design of later arrays as experiments proved that some probes or probe sets generated misleading data. This issue will be amplified with examples in section 3.3.

3.2.5 CDF files

Chip Definition Files (CDFs) describe the layout for Affymetrix GeneChip arrays. All probe set names within an array are unique. There are two versions of CDFs, one in ASCII text format and one in binary format. The information held in CDF files includes the name of the array, the ChipType, the number of rows and columns on the array, the number and type of Quality Control (QC) units and further Chip Reference material. More detailed

information about the contents of CDFs is given in Appendix B. Modified CDFs are used in this work to calculate the effect of removing certain probes from the data being analyzed, for example in section 3.6.2.

3.2.6 CEL files

Biologists or life scientists generally perform a series of experiments to compare tissues from different conditions, typically including a control. The results of each microarray experiment are stored in a **CEL file**, which contains the data from a captured image of the scanned GeneChip array and includes the raw intensities for probes. See section 3.2.3 for the public availability of CEL files from microarray experiments.

In essence, the CEL file consists of headers, intensity values, masks, and outliers. The headers give information about the number of rows and columns in the array, any offsets applicable and brief details about the experiment. This includes the type of GeneChip, a short form of the experiment description and the date and time of processing. The mean and standard deviation of the intensity values are given for each row/column position of the array, together with the number of pixels that were used to define that intensity value. Any masks and outliers are detailed at the end of the file. CEL files can be in character ASCII format, or in binary format, depending on which version of software is used to process the experiment results. The size of character CEL files can range between 11MBytes and 13MBytes. The size of binary CEL files is approximately 4MBytes. The detailed format of both ASCII and binary forms of CEL files are given in Appendix A.

3.3 Calculating expression measures

GeneChips measure expression by means of groups of 25mer probes. Establishing how to calculate an expression measure from these probe sets has an impact on which genes are shown to be differentially expressed between two conditions. One must also consider

the impact of post-transcriptional processing of RNA, particularly alternative splicing and alternative polyadenylation (see section 2.6) because Stalteri and Harrison [37] showed that probe sets representing the same gene did not necessarily give similar expression data.

In addition, due to biophysical issues such as probe-probe interactions and optical effects in the image capture and processing, one sees that some probes are not effective at measuring gene expression. The causes of outliers in large surveys of GeneChip data have been identified in some cases by Upton *et al.* [32]. It has also been established by Shanahan *et al.* [38], in the commonly used GeneChip HG_U133A for *H. sapiens*, that a clear bias exists in correlations between gene transcripts because of probe-probe interactions. The work in this thesis concentrates on exploring this bias further.

The calculation of gene expression has multiple steps because each gene has to be represented on an array by multiple probes, arranged in a probe set. Affymetrix designed their GeneChips to use probe sets and provided standard methods of microarray processing, including the Affymetrix MicroArray Suite (MAS). The steps involved in processing the raw data from microarrays to determine gene expression will now be described.

3.3.1 Microarray preprocessing

The procedural steps required to obtain a single composite expression value for each gene or probe set of a microarray is known as microarray preprocessing. There are commonly three steps involved: (a) background correction, (b) normalization and (c) summarization. These steps have been described and various methods compared in the literature, for example by Irizarry *et al.* [39] who produced the RMA pipeline (see section 3.3.3) and by Giorgi *et al.* [40] who benchmarked the three most used preprocessing procedures: MAS5, RMA and GCRMA, in the context of inter-array correlation analysis. These pipelines will be described and an additional one used in this work which was introduced by Affymetrix, called PLIER (Probe Logarithmic Intensity ERror estimate).

3.3.1.1 Background correction

The first step that is generally used to correct the intensity values of the microarray for each probe is background correction. The types of noise that might occur have been explored in section 3.2.1. In order to avoid these problems different preprocessing pipelines use different background correction algorithms while some pipelines even ignore this step altogether, such as FARMS (Factor Analysis for Robust Microarray Summarization) [41].

3.3.1.2 Normalization

Normalization is the process of adjusting the values of a number of sets of figures so that they can be compared on the same scale. In the context of microarrays, normalization can be described as the attempt to compensate for systematic technical differences between chips, so that the interesting biological differences between samples can be seen more clearly. Any differences in the treatment of two samples, especially in labelling and in hybridization, can bias the relative measures on any two chips. For their 133 series chips, Affymetrix introduced a new approach using a set of 100 "housekeeping genes" [42]. In processing, the chips are re-scaled so that the average values of these housekeeping genes are equal across all chips. This has proved better than using a single housekeeping gene, and is likely to be adequate for the majority of chips in practice.

Quantile normalization is a technique developed to make two or more distributions identical in statistical properties (mean and standard deviation), so that their values can be compared to each other. In the processing of microarrays Bolstad *et al.* [43] described the use of quantile normalization in making a valid comparison across all microarrays in the same experiment. The goal of the method is to ensure that the overall distribution of the corrected intensities is the same over all the microarrays in the experiment. The quantile normalization algorithm [43] consists of the following steps:-

1. Given n arrays of length p, form matrix X of dimension $p \times n$ where each array is a column

2. Sort each column of X to give X_{sort}

3. Take the means across rows of X_{sort} and assign this mean to each element in the row to get X'_{sort}

4. Get $X_{normalized}$ by rearranging each column of X'_{sort} to have the same ordering as original X.

The algorithm is applied simultaneously over all the data on each array.

3.3.1.3 Summarization

The third part of microarray preprocessing called summarization is the final part of the pipeline to produce expression data from the GeneChips. It is necessary to summarize the probeset data because in Affymetrix microarrays, multiple probes for the same transcript are condensed into a single representative signal. Commonly used approaches for the summarization are Tukey's Biweight (used by MAS5) and Median Polish (used by RMA). Tukey's Biweight is a weighted average of the particular log2 probe intensities, which down-weights probes which are more distant from the median of the probeset [44]. This approach should be more tolerant of outlier probes or spots than an average which is unweighted. Median Polish is used by RMA to estimate the model parameters for each probeset across all chips, having fitted a linear additive model of *signal + probe-affinity + error terms*.

In this work the two pipelines RMA and PLIER which produce the summarized expression values for particular genes are used. These pipelines will be discussed in the next three sections along with MAS5 which is the standard processing suite provided by Affymetrix.

3.3.2 MAS5

The Affymetrix MicroArray Suite (MAS) is a suite of programs used to calculate and store the raw data from microarray experiments in CEL files (see section 3.2.6 on page 45). MAS4 was the standard until January 2002 [45]. It calculated a weighted average of the probe-pair

differences (PM-MM) for each probe pair representing a gene. MAS5 was developed to improve on MAS4 [46] in two important ways. Firstly the intensities are calculated on a logarithmic scale before the average is taken; this equalises the contribution of different probes. Secondly an estimate of background based on MM replaces the MM itself in the difference PM-MM. This estimate is itself a weighted average of log probe pair differences. The Affymetrix paper "Statistical Algorithm Description Document" [44] gives further details. Giorgi *et al.* [40] point out that "MAS5 uses a single-array summarization technique (a robust Tukey-biweight average of the probe values) which treats each sample separately" unlike RMA and its sequel GCRMA (see next section) which can both introduce an artificial inter-array correlation.

3.3.3 RMA and GCRMA

It has been mentioned that the signal strength for MM probes can frequently be larger than that of the PM probes, which implies that the MM probe is detecting the true signal as well as background signal. This can result in unrealistic negative expression values, Kothapalli *et al.* [47]. The Robust Multi-Array Averaging (RMA) method of microarray normalization has been developed to background correct, normalize and summarize the probe level information without the use of the data obtained in the MM probes [1]. It is one of the most commonly used normalization pipelines. The background correction is effected by correcting the measured intensity of each probe in a given array by modelling all the data for the array as the product of two distributions (Gaussian and Exponential) that represent the noise and signal, respectively. The component representing noise is then subtracted from the measured intensity. The quantile normalization procedure (section 3.3.1.2) is carried out to ensure that the intensity values can be compared across all microarrays in the experiment. A final estimate of the summarized value from all the probes in each probe set is calculated by modelling the corrected and normalized intensities with a linear model. This model includes a noise component and a probe effect component in addition to the summarized

value for each probe set in a particular array. Tukey's median polish is used to estimate these parameters (the median polish is an exploratory data analysis procedure proposed by the statistician John Tukey [48]).

GCRMA (Guanine Cytosine Robust Multi-Array Averaging), is an improved form of RMA that uses the sequence-specific affinities of the probes to obtain more accurate gene expression values. It adjusts for background intensities in Affymetrix array data which include optical noise and non-specific binding. The background adjusted probe intensities are then converted to expression measures using the same normalization and summarization methods as RMA. The sequence information is summarized in a more complex way than the simple GC content. Instead, the base types (A, T, G or C) at each position (1 - 25) along the probe determine the affinity of each probe. The MM (mismatch) data is not lost; the user has a choice in version 2 of GCRMA to use either MM probes, a user-defined list of negative control probes or the package internal non-specific binding data.

3.3.4 PLIER

Affymetrix developed an algorithm called PLIER [2] to improve the gene expression value (a summary value for a probe set) for the GeneChip microarray platform compared with the MAS suite. The improvement in signal value is achieved by accounting for experimentally observed patterns for feature behaviour and by handling error appropriately at low and high abundance. It will be shown later in this work that PLIER can also improve the gene signal compared with that calculated by the RMA algorithm. PLIER is designed to produce an improved signal by accounting for experimentally observed patterns in feature behaviour and also by handling error at appropriately low and high signal values.

The PLIER algorithm introduces a probe affinity parameter, which represents the strength of a signal which is produced at a specific concentration for a particular probe. The probe affinities are calculated using all the data for an experiment across its arrays. The error model used assumes that error is proportional to the recorded intensity values, rather than to

background-subtracted intensity.

3.4 Data validation

It is important to have confidence in the methods used to calculate gene expression from the raw data of microarrays. Some of the commonly used methods of calculating a single measure of gene expression for a probe set e.g. MAS5, RMA, GCRMA and PLIER, have been described. Some of the work which has been done to validate the data produced for gene expression from microarrays will now be explored. Spike-in controls introduced by Affymetrix will be examined. Then methods developed in Dr Andrew Harrison's group at the University of Essex to improve the results of processing microarrray data: unique mappings and correlation heatmaps, will be explained.

3.4.1 Spike-in and Affycomp

Spike-in controls or spikes are external reference RNAs which can be used to improve the quality of microarray data. The use of spikes allows several parameters of the platform quality to be evaluated, for example, the sensitivity and specificity of the microarray experiments, the accuracy and reproducibility of the measurements and the assessment of technical variability which may have been introduced by the labeling procedure, hybridization and image scanning [49]. Spike-in HG_U95 experiment and HG_U133 experiment CEL files were provided for comparison analysis on a website called Affycomp and over 50 different sets of results were gathered for different probe summary methods. These comparisons are now cited here in two ways: the first is an investigation of the PLIER algorithm by Therneau and Ballman and the second is a comparison of methods used on HG_U133A data.

The PLIER algorithm was investigated by Therneau and Ballman [50] who conclude that it improves the gene expression value compared with the MAS5 algorithm and captures the key characteristics of the ideal error function. The diagram that demonstrated their

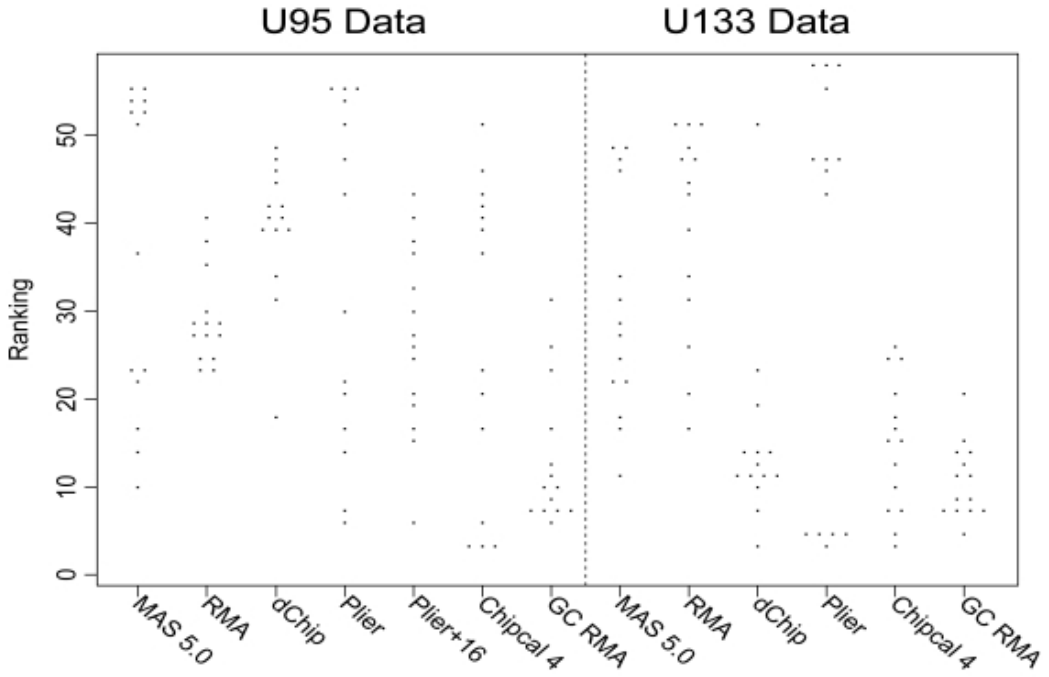


Figure 3.5: Rankings of selected methods on 14 outcomes, from the Affycomp website

findings is reproduced here as Figure 3.5. It shows the ranks for 14 measures of accuracy (concentrations) for the normalization and analysis contributions, of MAS5, two early methods: RMA and dChip, two more recent methods: chip calibration and GCRMA, and PLIER. Using the U133 data it shows consistently good rankings for GCRMA and Chipcal 4. It shows very good rankings for PLIER for five of the 14 outcomes in the U133 data. However the remaining nine are poor. The results for RMA are more widely spread across the rankings, but show no very good rankings and no very poor rankings. It can be concluded that although PLIER could be a better method to employ in some experiments, it must be used with caution because of its poor performance in other cases. The major finding of Therneau and Ballman [50] was that the model used by PLIER for its error correction algorithm possesses many of the key characteristics of the ideal error function for fitting individual probe calibration curves. This was despite PLIER using the counter-intuitive assumption that the error of the mismatch probe was the reciprocal of the error of the perfect match probe. More evidence of the workings of PLIER will be discussed among the results and findings of this project.

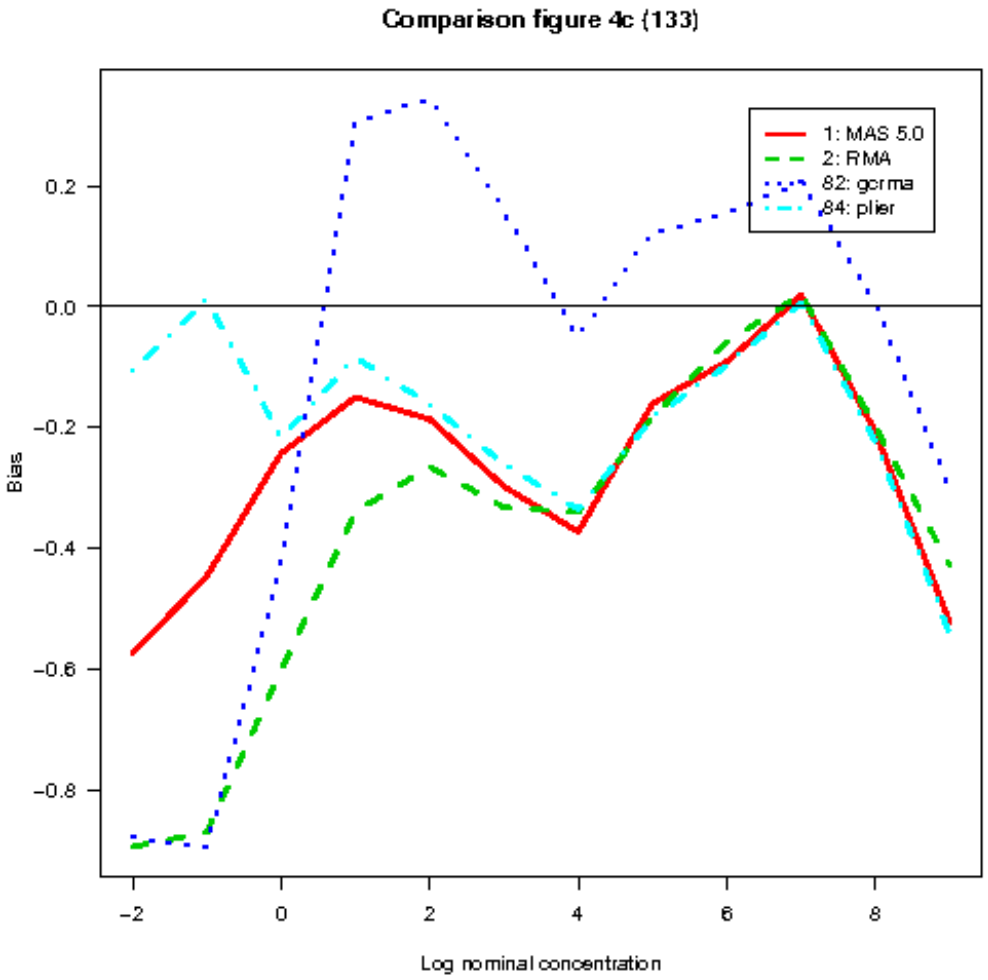


Figure 3.6: Comparison of Bias in four selected probe summary methods at different concentrations for U133 data

Figure 3.6 shows a comparison of probe summary methods using data from the Affycomp website. Local slopes are calculated which represent the expected observed log fold-change for probesets with true fold-change of 2 but they are presented as a function of the total nominal probeset concentration in the samples being compared. In theory the local slopes should be 1 so the bias (difference between the observed local slope and 1) is shown.

The figure shows major differences between the methods at low concentrations, but a convergence at higher concentrations with the exception of GCRMA, which achieves a different average bias across concentrations.

3.4.2 Generation of Unique Mappings

3.4.2.1 Introduction

It has been found that probe sequences can **partially** align to one or more locations in the genome other than the intended one. This is known as cross-hybridization [51]. Another effect called multiple targeting, is where probes can align **exactly** to more than one place in the genome. In order to minimise these effects, the generation of "unique mappings" has been developed for the Human genome by Dr Olivia Sanchez-Graillet et al. [51], and extended to other species such as Mouse, Arabidopsis and Rice. The unique mappings are the first stage in a pipeline to analyse CEL files (see section 3.2.6 on page 45 for more explanation), which includes the production of heatmaps using regression analysis. Although many probes may be discarded when filtering for unique mappings, the remaining probes can be used as more reliable measures of target expression. The next few subsections give an overview of the process.

3.4.2.2 Obtaining sequence file and probe data

Sequences of exons and transcripts are downloaded from the Ensembl database at EBI, Cambridge, via Biomart and stored in local servers as text files. BioMart is an OpenSource, query-oriented data management system. It can be used with any type of data and is

particularly suited for providing 'data mining' type searches of complex descriptive data. The sequences of probes are obtained from the Affymetrix website and stored in FASTA file format.

3.4.2.3 Megablast alignments

Megablast is a derivative of the sophisticated software package called BLAST, which is a Basic Local Alignment Search Tool [52], introduced in section 2.8.1. It is used for finding statistically significant similarities between sequences by evaluating alignments. *Blastn* compares nucleotide sequences to one another (hence the '*n*'). *Megablast* is similar to *blastn* but optimised to find near identities very quickly. It is much faster than the standard *blastn*. The probe sequences from Affymetrix are aligned to the exon and transcript sequences by using the *megablast* tool and the sequence files previously obtained. The files containing the mappings have to be filtered using *gawk*, to save processing time later.

3.4.2.4 Configuration files

The programs for generating unique mappings have been developed to allow for different versions of biological databases, different species, different file locations in the servers, different program versions and various types of arrays. These parameters are used as input data to the programs, specified by information in two files: a configuration file and an array file.

3.4.2.5 Auxiliary data

In order to obtain the unique mappings, it is necessary to generate initial data in tables in the database. The information required is:-

- Exons information
- Transcript information

- Mappings of probes to exons
- Mappings of probes to transcripts, which align with $values >= 20$, where *values* are the alignment_length $\times$ the id_percentage.
- Synonymous exons (exons in the same chromosomic region but with different identifiers)
- Probes which map to exon-junctions

Synonymous exons and exon-junctions are stored in the corresponding tables in the database as well as in text files.

3.4.2.6 Unique mappings identified

Once all the required data has been processed and stored in the database, the unique mappings are identified and the information is saved both in text files and in the database. A more detailed description of producing the unique mappings is given in Appendix C.

3.4.3 Correlation matrices

It is possible that within one probe set, subsets of probes could be measuring different exons, as shown by Stalteri and Harrison [37], and thus different transcripts because of biological signals such as alternative splicing (section 2.6). If all such probes were included in a study to see if probes were correlated with each other then the results could be misleading. Therefore the unique mappings of groups of probes, described in section 3.4.2, which map uniquely to the same exon were employed so that the correlation coefficients between pairs of these particular probes will not be affected by alternative splicing.

3.4.3.1 Correlation coefficients between pairs of probes

The data used here was 6685 HG_U33A CEL files which were downloaded from the NCBI GEO repository. After normalizing each CEL file the values of the correlation coefficients

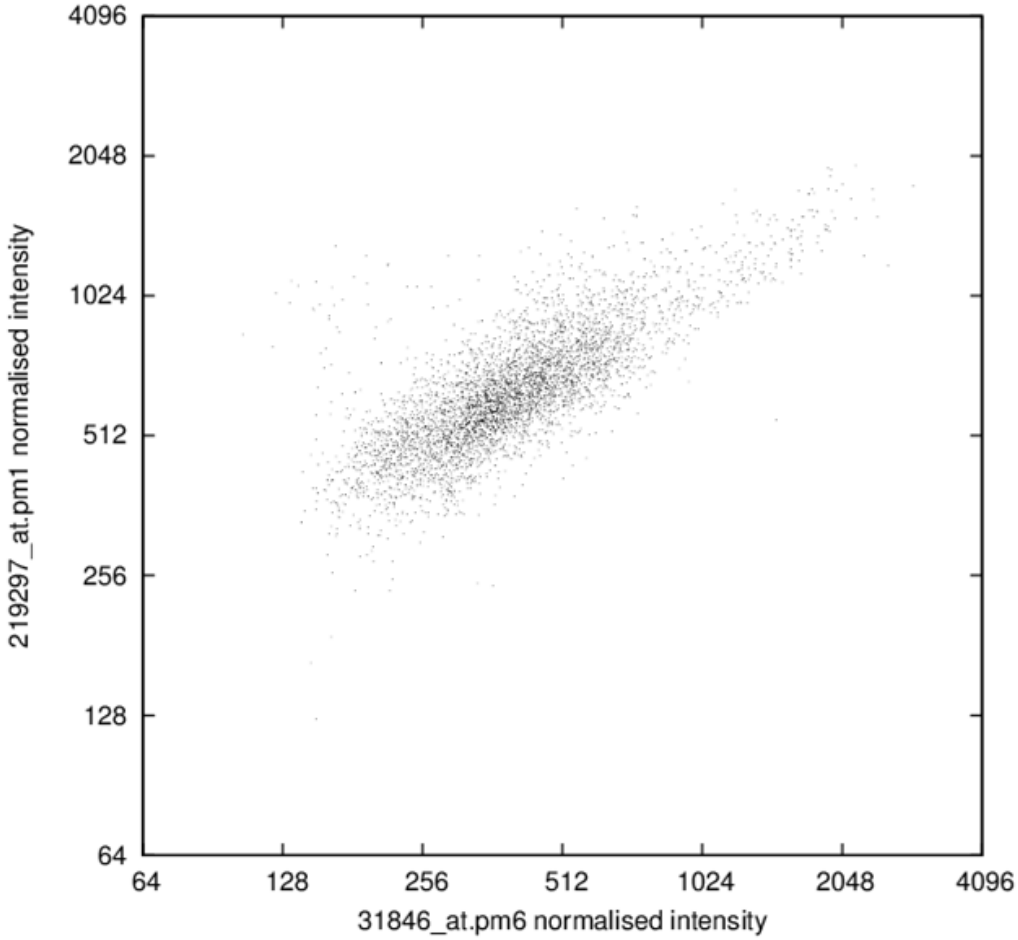


Figure 3.7: Scatter diagram comparing probe pm6 from probe set 31846_at with probe pm1 from probe set 219297_at (Source: Upton *et al.* [12])

between pairs of probes from within the same exon were examined to search for anomalies. The probe set 31846_at which is one of two sets designed to match the gene RHOD provides an example. This probe set contains 16 PM probes all drawn from the same exon. The value of the correlation coefficient between all pairs of these PM probes is strongly positive with the exception of probe pm6 (the sixth probe of the probe set), which has near zero values for its correlation coefficients with all the other probes. To test the hypothesis that probe pm6 is related to other probes from different exons, the value of its correlation coefficient with every other probe in the array was determined. It can be seen in the example scatter

plot in Figure 3.7 that the probe pm6 from probe set 31846_at is correlated with probe pm1 from probe set 219297_at because of the grouping of the plotted points in an upwards sloping direction. (The probe set 219297_at was designed to measure activity of the WDR44 gene.) The correlation coefficient in this case, r = 0.78. The reason for the correlation needed to be found, and Upton *et al.* [12] concluded it was because of both probes having runs of guanine. The two probes in this case both had sequences of guanines; probe pm6 from probe set 31846_at has five 'G's' at position 16 of the probe sequence (base sequence TCCTGGACTGAGAAAGGGGGTTCCT), and probe pm1 from probe set 219297_at has six 'G's' at position 1 of the probe sequence (GGGGGGATAGTCTTGTTTCTAGCTT).

3.4.3.2 Heatmaps

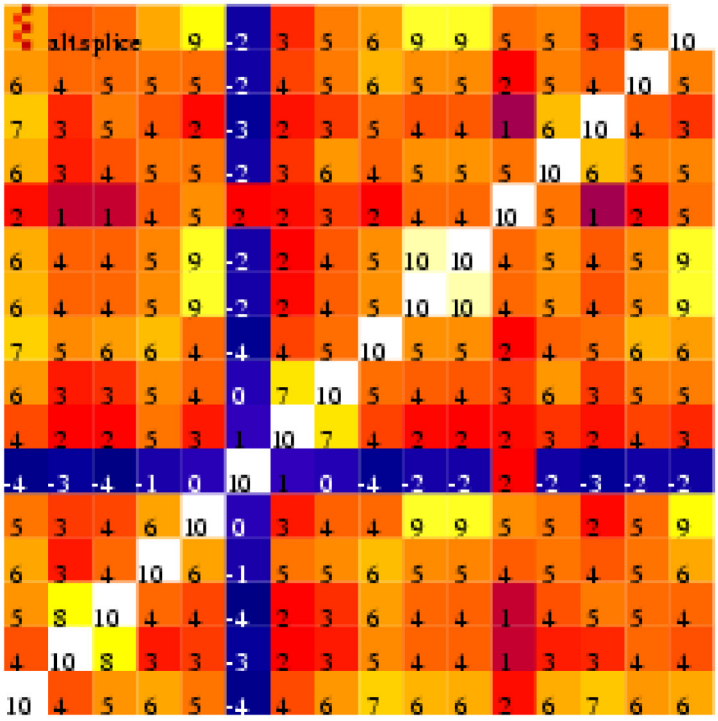


Figure 3.8: Correlation matrix showing the correlation coefficients between every pair of the 16 perfect match probes that form the 31846_at probe set (Source: Upton *et al.* [12])

Figure 3.8 shows a **heatmap** created by using only the 16 PM probes from probe set 31846_at, which as mentioned in the last section map uniquely to the same exon. The

heatmap is a visual display of the correlation coefficients between pairs of these probes. The values have been calculated, multiplied by 10 and rounded. This number is shown and the shade in each cell represents a value between 1 and 10 for positive correlations denoted by colours in the red to yellow and white spectrum. Negative correlations are shown by shades of blue becoming darker with higher negative numbers. The values on the main diagonal are all 10, since the correlation coefficient is 1 for a value compared with itself. Probe 6 is evidently behaving in a different way with its blue colour, showing that it is not correlated with the other probes as would be expected. Upton *et al.* [12] looked at other heatmaps and found that such 'misbehaving' probes were not unusual. The base sequences were looked at and the observation made that a frequent feature was a sequence of four or more guanines. It had already been noted by Wu *et al.* [53] that probes containing runs of guanine were typically poorly correlated with others in their probeset. Upton *et al.* [12] showed that such outlier probes are well correlated with each other.

This work concentrates on the effects that runs of guanine bases have on the gene expression data obtained from microarrays. The chemistry of runs of guanine bases and how they can affect structures within DNA strands or loops will be explained.

3.5 G-quadruplexes, G-stack and C-stack probes

Han and Hurley [54] have described how four guanines (G's) from various places in a DNA sequence can form a planar structure, termed a **G-tetrad** or a **G-quartet**, through Hoogsteen hydrogen bonding (see the schematic Figure 3.9(a)). They describe diagram (b) in the same figure as a tetramolecular parallel **G-quadruplex**, which is formed from four DNA strands each with a single G-rich repeat. G-quadruplexes can also be formed by a DNA strand or strands which loop in ways shown by diagrams (c) and (d). Each quadruplex is made up of G-tetrads which are indicated by the planar squares in (b), (c), and (d).

Burge *et.al.* have described a variety of G-quadruplexes formed from both telomeric

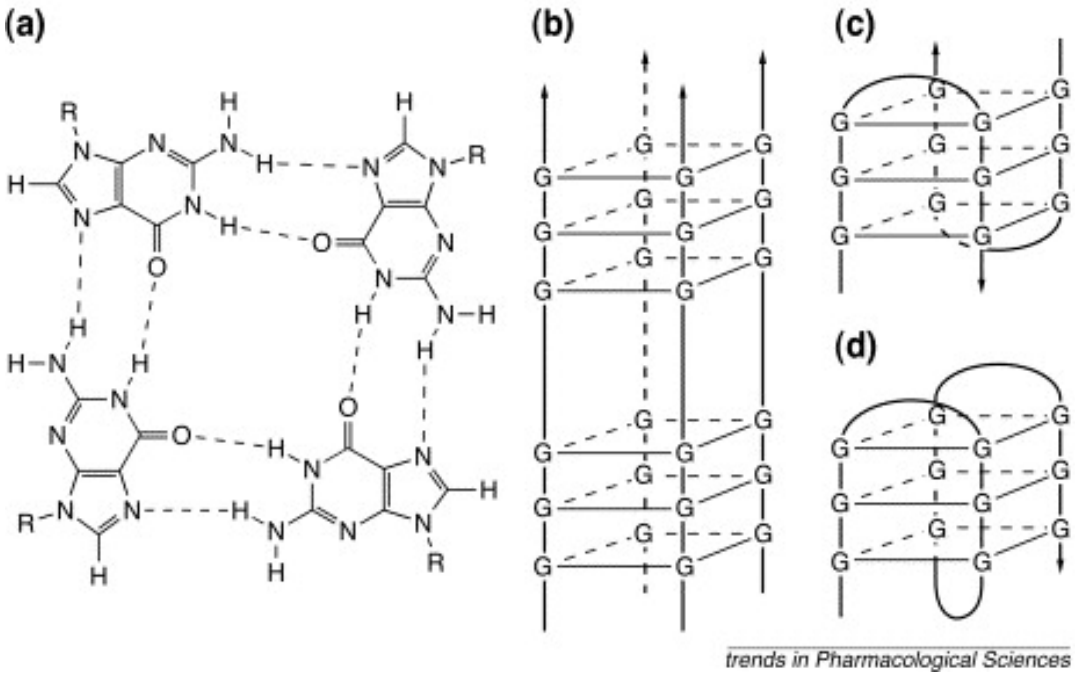


Figure 3.9: Schematic of a G-tetrad and G-quadruplexes. (a) Four guanine (G) residues can form a planar structure, termed a G-tetrad, through Hoogsteen hydrogen bonding. (b) A tetramolecular parallel G-quadruplex can be formed from four DNA strands each with a single G-rich repeat. (c) DNA sequences that contain two or more G-rich repeats can form GG hairpins, which in turn dimerize to form several types of stable bimolecular quadruplexes, termed intermolecular antiparallel G-quadruplexes. (d) DNA sequences with either four G-rich repeats or long G tracts can fold upon themselves to form an antiparallel intramolecular quadruplex, termed an intramolecular foldover G-quadruplex. (Source: Han and Hurley [54])

and non-telomeric DNA sequences [55]. Their survey included potential quadruplexes that can be formed from G-rich sequences which are present within a wide range of genes and in non-coding regions of many genomes. These structures are of particular interest as possible targets for therapeutic intervention. It is noted that quadruplex sequences occur in the promoter regions of several cancer genes, Burge *et.al.* [55]. With a great deal being discovered and much still unknown about guanine rich sequences, it was decided to concentrate on surveying the extent and potential effects of such sequences in the microarray data available in public repositories.

Langdon *et al.* [56] have discussed the molecular processes occurring on the surfaces of GeneChips due to the high surface density of probes. Interactions between next-door probes affect their rate and strength of hybridization to targets. Targets may partially bind

to more than one probe, and competing targets may hybridize to the same probe. These partial hybrids can result in some probes not reaching thermodynamic equilibrium during hybridization. Some probes may fold, or some targets may fold up or cross-hybridize to other targets. Having surveyed many experiments searching for systematic effects such as these, Langdon *et al.* discovered a family of thousands of probes which show correlated expression across thousands of GeneChip experments. These probes contain runs of guanines, suggesting that G-quadruplexes are able to form on GeneChips which results in bias when averaging signals from multiple probes.

A **G-stack probe** is defined for this work as being a probe with one or more sequences of exactly four G's, i.e. GGGG, a run of four guanine bases. In similar fashion, a **C-stack** probe is defined as being a probe with one or more sequences of exactly four C's, i.e. CCCC, a run of four cytosine bases. These definitions will be used in describing the analyses. The C-stacks will be used as a control in some of the analyses of G-stacks, in a similar way to that in which Shanahan *et al.* [38] used C-stacks as a control when analysing the effect of G-stacks.

3.6 Three types of analysis used in this work

The analysis of the effect of G-stacks in microarray probes is the focus of this research. Three types of analysis on microarray data are used. They are described in this section and can be briefly called:-

1. **Average correlation of expression levels over all CEL files of an experiment using a sample of 1000 G-stack probes**

   This analysis uses an estimate for the size of bias that G-stacks introduce across experiments, as introduced by Shanahan *et al.* [38]. As a control, the same analysis is done using C-stacks, which have been shown not to introduce a bias in microarray experiments.

2. **Summarized expression data**

   This analysis calculates the median difference of expression data between microarray data with G-stacks left in and microarray data with G-stack probes removed.

3. **Analysis where the correlation between pairs of probe sets is greater than 0.4**

   This analysis compares every probe set to every other probe set and takes the results where the correlation is greater than 0.4 to see the most significant results.

These analyses all use the RMA normalization method in the first instance. Later studies will compare the results of using the PLIER normalization method with the results of using the RMA normalization method.

In the following explanations of the analyses the GeneChip used for the examples is HG_U133A, a human chip which has gained widespread acceptance among researchers.

3.6.1 Average correlation of expression levels

It has been established by Upton *et al.* [12] that the G-stack bias is variable across individual experiments. A proxy is therefore used to estimate the size of the bias for other experiments. Computation for the proxy is as follows: a sample of 1000 G-stack probes are randomly selected from probe sets with only one G-stack probe in them. This set of probes is referred to as G_r. For each of the 176 HG_U133A experiments deposited at GEO (the same selection that has been used in the referenced analyses Upton *et al.* [12] and Memon *et al.* [57]), the following was computed

$$n_i^{[\alpha]} = \log(e_i^{[\alpha]}) - R^{[\alpha]}, \qquad i \epsilon G_r,$$

where $[\alpha]$ represents an individual CEL file, $e_i^{[\alpha]}$ is the expression level for the *i*-th probe of CEL file $[\alpha]$ and $R^{[\alpha]}$ is the average of the log of the expression levels over all non-control probes (i.e. the probes which are biologically relevant) for that CEL file. The correlation of $n_i^{[\alpha]}$ over all CEL files is computed and the average correlation ρ_{ij} for all $i \neq j$ is calculated.

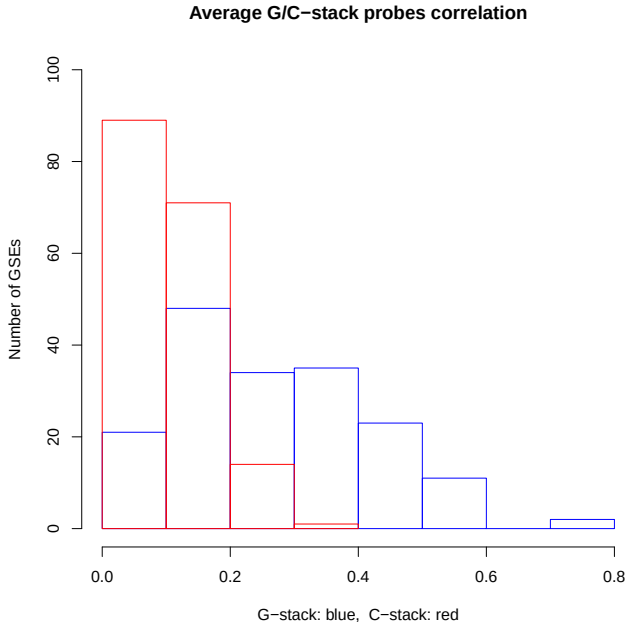


Figure 3.10: Bar chart of the average correlation between G-stack probes (in blue) and C-stack probes (in red) for 176 HG_U133A GeneChip experiments deposited at GEO.

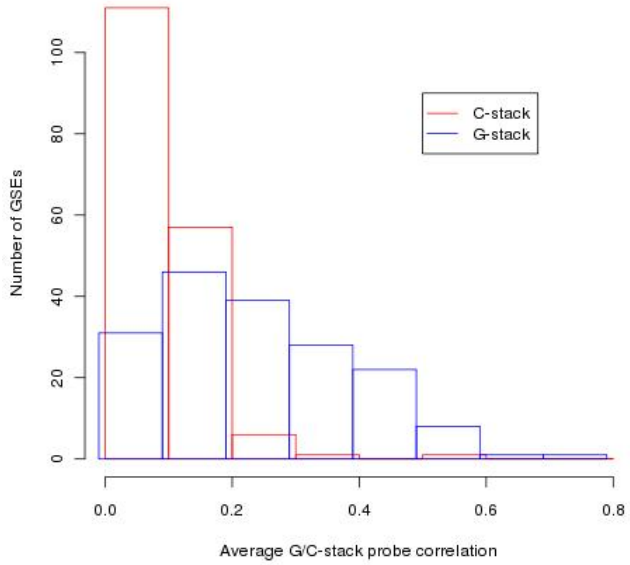


Figure 3.11: Bar chart of the average correlation between G-stack and C-stack probes for 176 HG_U133A GeneChip experiments deposited at GEO. Reproduced from the paper by Shanahan *et al.* [38]

A control was needed, so similarly defined average correlations for each of the same experiments were computed using 1000 randomly selected probes with runs of four cytosines. The bar chart of the resulting 176 average correlations for both G-stacks and C-stacks is shown in Figure 3.10. It can be seen that the average correlation for the C-stacks is much closer to zero. The chart is seen to be similar to Figure 6 in the paper by Shanahan *et al.* [38] which is reproduced here as Figure 3.11. The differences are explained by a different random set of 1000 G-stack and C-stack probes being selected.

3.6.2 Summarized expression data

The Affymetrix GeneChip HG_U133A was the first microarray to be used by Upton *et al.* [12] to investigate the effect of G-stacks on normalized data. The GeneChip HG_U133A contains a total of 22,283 annotated probe sets. Table 3.1 shows how many probes sets have various numbers of G-stack probes in them. This information is relevant to the analysis of the effect that G-stack probes have on expression level data for an experiment. It is noted that more than one-third of the probe sets contain at least one G-stack probe, and that 10% of the probe sets contain two or more G-stack probes.

No. of G-stack probes in a probe set	**0**	**1**	**2**	**3**	**≥4**
No. of affected probe sets	13,985	5188	2124	667	319

Table 3.1: The numbers of probe sets in HG_U133A that have particular numbers of G-stack probes

The dark columns in Figure 3.12 show the median difference of expression data between microarray data with G-stacks left in and microarray data with G-stacks removed. The light coloured columns show the same calculations for C-stacks. To obtain these calculations, all the probe sets from the HG_U133A GeneChip were analysed and classified in the following way:-

- $g < n >$ is the list of probe sets with $< n >$ G-stack probes in them, so that for

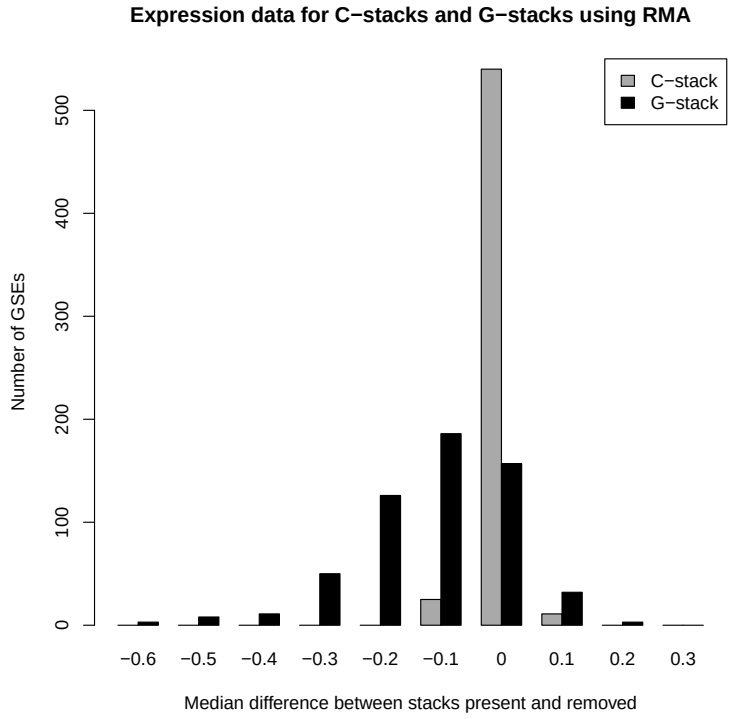


Figure 3.12: HG_U133A expression estimates showing median differences when C-stacks and G-stacks are removed respectively

example, g3 is the list of probe sets with 3 G-stack probes in them.

The expression data from all the CEL files of each GSE was summarized in Figure 3.12 by using the following process:-

- expression data was normalised with the Bioconductor package called *affy* using the RMA method
- using a modified CDF (see section 3.2.5), the data was normalised again with G stack probes from groups g2 to g6 removed
- for each probe set in lists g2 to g6, the difference in the normalised expression data between having G-stack probes left in and having G-stack probes removed, was calculated
- the various quantiles between this difference and the expression levels of the original data with G-stack probes left in were calculated
- all the above steps were repeated using C-stacks in place of G-stacks
- the median differences for both G-stacks and for C-stacks removed were plotted as a bar chart showing the frequency of the median values for all the GSEs (Figure 3.12)

If there were no bias in the data due to the G-stacks, then Figure 3.12 would show the median difference values for the G-stack data more evenly distributed around zero. However, it can be seen that most of the median difference values are less than zero, which confirms the anticipated bias effect. For the C-stacks, the median difference values are grouped much closer to zero, showing that C-stacks do not exhibit a bias.

3.6.3 Analysis of GT correlations

It is expected that data from probes within the same probe set would be correlated with each other because they should be regulated by the activity of the same gene. However, this work

is testing the extent to which G-stack probe sets are correlated with other G-stack probe sets. It was decided to compare the extent to which C-stack probe sets are correlated to other C-stack probe sets, as they were shown by Shanahan *et al.* [38] to be a good control.

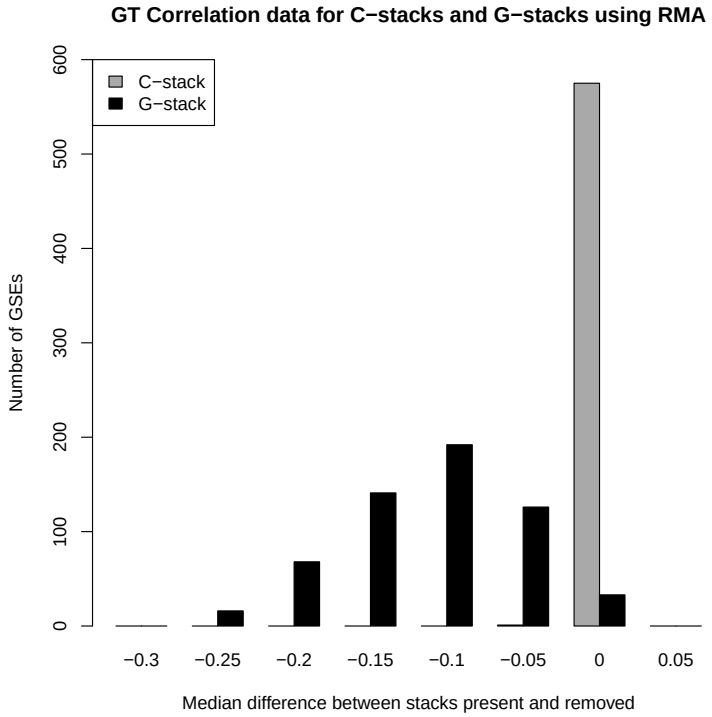


Figure 3.13: HG_U133A GT correlation data for median differences of C-stacks and G-stacks present and removed respectively

In this analysis every probe set is compared to every other probe set, and the results taken where the correlation is greater than 0.4 to see the significance. From here onwards where correlations have been taken as greater than 0.4 they will be referred to as **GT** correlations. Figure 3.13 shows that the control figures for the correlations of the C-stack probes are all grouped near zero, whereas the figures for the G-stack correlations are all negative and spread much more widely. These results confirm that C-stack probes are not correlated with C-stack probes from other probe sets. However, G-stack probes are showing some correlation with other G-stack probes from different probe sets.

Correlation data was also looked at to determine if correlations of less than 0.4 between

G-stack probe sets were significant. These will be referred to henceforth as **LT** correlations. Figure 3.14 shows the extent of LT correlation (i.e. the median differences where the correlation is less than 0.4). The C-stacks are heavily grouped around zero again which shows that this measure is not significant for C-stacks. The G-stack LT correlation data are more spread out, showing that there is some significance in this measure for G-stacks, especially as compared with the control C-stacks.

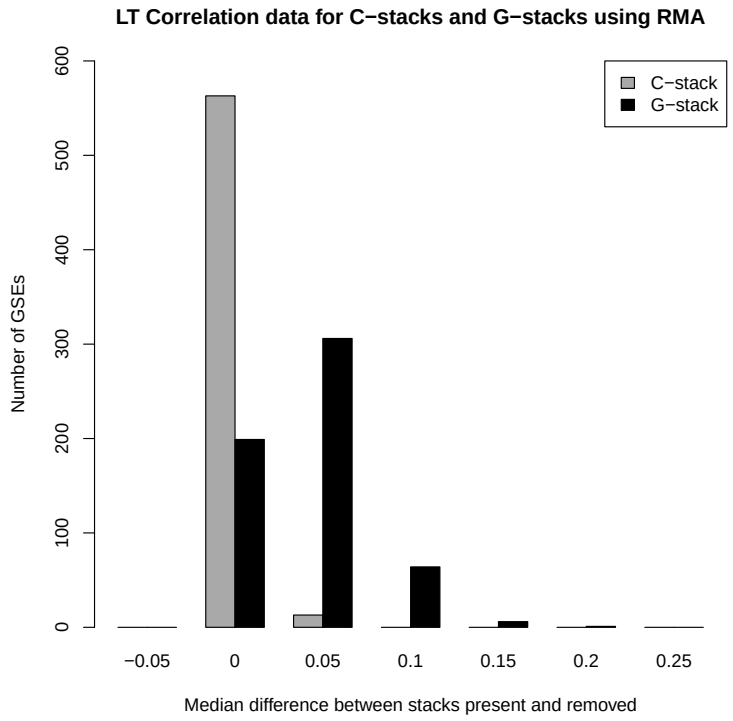


Figure 3.14: HG_U133A LT correlation data for median differences of C-stacks and G-stacks present and removed removed respectively

3.7 Conclusion

This chapter has explained the technology and use of microarray data. The preprocessing techniques, normalization and summarization methods of processing CEL file data have briefly been explored, with their relevance to the measurement of gene expression. The generation of unique mappings to improve the quality of probe selection has been explained.

This led on to the creation of correlation matrices and the visualization of these in heatmaps, which highlighted certain probes which were more highly correlated with each other than with their designed RNA sequences. Then G-quadruplexes and G-stacks have been defined with their significance in interpreting microarray experiment data. Finally some analysis methods have been outlined which will be used later in this work.

Attention is now turned to the computing resources which can be used to carry out wide scale analysis of microarray data. Grids and clouds have been introduced. The next chapter will focus on experience gained in using a grid.

Chapter 4

Grid Computing: The National Grid Service (NGS)

This chapter will describe a few examples of grids which have been used by the academic community and then concentrate on the National Grid Service (NGS) and some experience gained by using it. The NGS was the first service tried by this project for some bioinformatics computation outside the University of Essex. It offered a free computing resource to academic researchers in the U.K. The name of the NGS was changed to NES (National E-infrastructure Service) when the NGS and the separately funded GOSC (Grid Operations Support Centre) merged. In this chapter, the name NGS will be used, although it refers equally to the NES which may have been the correct name in later years.

4.1 A few examples of Grids

The GLOBUS Project [24] with its associated software, the Globus Toolkit [58], has been key in developing fundamental techniques which are needed to build computational grids. The challenge is to enable software applications to integrate instruments, displays and computational resources that belong to diverse organisations in widespread locations. The project has included investigations of security, communication protocols and data

management mechanisms as well as resource management. The NGS services were based on the Globus Toolkit offerings for job submission.

Condor [59] is a high throughput computing environment that can manage a large collection of diversely owned machines and networks. It is well known for harnessing idle computers with its cycle stealing technology, but it can also be configured to share resources. Its environment has a layered architecture and can support both sequential and parallel applications. Its middleware is open-source grid software which allows users to submit jobs in a reliable fashion to remote grids and batch systems such as GLOBUS, and Unicore.

Unicore (UNiform Interface to COmputing REsources) is a ready-to-run grid system of middleware that aims to make distributed computing and data resources available in a seamless and secure way. It implements several open standards to allow different networks to communicate with each other, and integrates strong security and workflow capabilities. Its website provides more details of its objectives: *http://www.unicore.eu/unicore/*.

4.2 Experience of using the NGS in 2009

4.2.1 Characteristics and Facilities of NGS

The U.K. National Grid Service had 25 member institutions in 2009. Eleven were partners and 18 were affiliates. There were over 900 registered users of whom over 250 were active, according to figures given at the IF09 Innovation Forum held in London in October 2009. There were over 20 projects actively using databases and over 2000 active user certificates.

The NGS e-infrastructure was connected via the JANET network to HEIs (Higher Education Institutes), GridPP (a particle physics Grid), and JISC (*www.jisc.ac.uk*) among other networks including at least one European Grid, the EGI (European Grid Initiative).

The NGS offered:-

- Complete services

- Data services
- Access services
- Central support services

It also offered a Hermes data client for drag and drop file movements, and for large data set requirements.

The NGS promoted open standards as vital for Grid development. It enabled various levels of resource exchanging, depending on the type of organisational membership which HEIs and others required. NGS also encouraged a Campus Champion to be appointed at each member institution who would know the resources available with NGS and propagate information. A website and magazine-style emails gave information about how to join and use the NGS, as well as details of projects which were successfully using the Grid service.

The NGS project had been seeded with four large computers spread among founding member institutions, and other computer resources were offered by members. HECToR, the High Performance Computing project at Edinburgh university was available to researchers through the NGS. The NGS services were free to academic research users in 2009, though partner institutions were working out what it cost them to offer services, and plans to charge for services in the future were being discussed.

4.2.2 Goal of the NGS

The goal of the NGS was to deliver a production quality national e-infrastructure in support of academic research across all Higher Education Institutes (HEIs) in the UK. This enabled coherent electronic access for UK-based researchers to a variety of computational resources and facilities, independent of the researcher location.

4.2.3 How to run jobs on the NGS

The initial task for users is to analyse their computing requirements and decide if their need is for high performance or for high throughput. High performance jobs require cutting edge supercomputing solutions such as that offered by HECToR at Edinburgh University. High throughput jobs typically need to run large numbers of small programs, and need careful management of their input and output data.

The NGS had at least three ways of submitting jobs to give users a degree of flexibility in the way they interacted with the grid:-

- Command line entry via Secure Shell (SSH) or a variant of SSH. This needs the client to have installed some software such as SSH which may be complex for some users. It can be flexible and efficient when working. It also needs the GSI (Grid Security Infrastructure) software installed so that security of access can be managed.

- via the API which is an Application Programming Interface that uses a batch job submission approach

- Portal template, for example, Application Repository or via WMS (Workload Management System). A portal gives the user entry via a web page into which they type their username and password. When the user has been checked for authorization, the portal then gives the user access to a list of application groups, examples and tutorials. There are sample job templates which can be modified. One can browse files and transfer them. After a job has been executed, the output can be retrieved.

4.2.4 Software available on the NGS

Some bioinformatics software was already installed and available to use. BLAST was available for nucleotide sequence searches, and Matlab for simulations or algorithms. Taverna had graphical tools available for WMS (*www.taverna.org.uk*). The Portable Batch System (PBS) was available for job scheduling.

At the NGS Innovation Form in October, 2009, it was made clear that the NGS was not currently administered systems-wise fully as a Grid. Its documentation was not complete, it had uneven loads and could run out of quota on a node (a machine within the grid), and its nodes were not always in synchronization with each other. It was apparent that one could use nodes on the grid that had the software installed that one needed, but that the service levels would be unpredictable and patchy.

There was little future funding predicted for the NGS. Some confidence was being placed in the future European Grid Infrastructure, an Enabling Grid for E-Science (EGEE - III project), which is discussed further in section 4.2.6. Each national Grid service in European countries was collaborating to further the project, together with Research Communities. Amsterdam had been chosen as the central location for the EGI, with 25 core staff and around 25 staff cooperating from their NGIs. It was planned that the project would not write software but install it from others' projects using standards and open source from the Open Grid Infrastructure (OGI). User support and services would be provided via a federated Helpdesk. Virtual Research Communities such as the Particle Physics community were in the forefront of recommending the research infrastructures which would be sustainable beyond a single funded project. The EGEE would sponsor training events, an applications registry (to find one's required applications on the EGI) and a Respect program to identify software which works well with the Grid. Two annual conferences were envisaged: a European Grid conference and a conference for Users of European E-infrastructure.

4.2.5 Benefits of the NGS to the researcher

There were reasons for academic institutions to belong to the NGS as a member, or at least to be affiliated. Systems administration support was available, user training materials could be shared and the grid was able to supply top-up computing resources for the institution.

The benefits of the NGS to individual researchers included the following:-

- free access to computing resources outside one's institution

- standard accounting framework for authorization and certification
- standard interfaces to enable submission and running of jobs
- community practitioners were being identified to share excellence
- data management standards were propagated
- data services such as Oracle and MySQL were available
- Workflow Management System available to aid users in submitting jobs and browsing their status
- Portal system to provide web page access to job submission and monitoring
- specialist services such as access to HECToR in Edinburgh, and large scale visualization at Rutherford Appleton Laboratory
- access to an international gateway for wider Grid participation

4.2.6 Wider Horizons

The NGS provided core services and support activities to facilitate collaborative access to computing and data resources in support of UK researchers. Integrated with partner infrastructures in Europe (through the project called Enabling Grids for E-scienceE (EGEE) and the European Grid Infrastructure (EGI)), the USA, and elsewhere in the world, the NGS ensured that UK researchers can effectively and efficiently exploit facilities and collaborations across the world.

E-Infrastructure provision has been identified by the national ministers in the EU Competitiveness Council as crucial to the future leadership position of EU economies in a global market place. Over the past several years the NGS has established a leadership position in e-infrastructure provision for the UK internationally.

4.2.7 Institutions which hosted Computer Resources

Partner sites involved with the NGS were:-

- Cardiff University
- University of Edinburgh
- University of Glascow
- Lancaster University
- University of Manchester
- University of Oxford
- Queen's University Belfast
- University of Leeds
- Rutherford Appleton Laboratory
- University of Westminster

Other sites were 'affiliated' to the NGS and provided resources to the NGS, but with more conditions. Members of the GridPP collaboration also offered resources.

4.2.8 Using R and Bioconductor on the NGS

There were only six institutions, or nodes of the grid, which offered R as a working software platform. None of these had Bioconductor modules installed. Upon enquiry, it was found that users can install extra R modules for themselves in their own disk file area, or request that extra modules be installed via the Helpdesk. For the latter option to be successful, it would need to be demonstrated that more than two or three users required to use the extra modules.

It transpired that R was not used by many researchers on the NGS. The most used program was compiled MATLAB. The biggest CPU time user was NAMD, a parallel molecular dynamics program which is used for high performance simulation of large biomolecular systems [60].

4.3 Reflection on the experience of using NGS

It is valuable to have the opportunity of using free external computing services available directly from one's own desk and local PC. The inital joining of the NGS to allow credentials to be recognised and accepted took little more than one week. The process of loading up additional R modules, writing appropriate Job Description Language to launch batch jobs, and debugging initial submissions meant that getting an R script to be successfully executed in batch mode with its data files took one month.

4.3.1 Joining the NGS as a new user

Fortunately the University of Essex, although neither a partner nor an affiliate site, had a staff member who was authorised to sign up new users to the NGS. The first requirement was to obtain a digital certificate from the UK Certification Authority and load this into a PC browser. The online application form to apply for an NGS account was then usable. The form required details of the scientific research planned and the reason for wanting to use the NGS. All applications were then subjected to a light-weight peer review process that could take one to two weeks. In the case of this study the review took only one week. The final part of the process was to visit the Registration Authority (the designated Essex university staff member) with photographic id and the NGS documentation to obtain a signature of approval.

4.3.2 First use of the NGS

The User Certificate which had been obtained from the Certification Authority was stored on the local drive with a 20-digit password. The first problem was that this did not work on the next stage. Apparently one needs a password of seven characters or less for it to work in the next stage of software. The certificate had to be exported from the local browser and re-imported with a shorter password.

4.3.3 Private keys and Public keys

A digital certificate contains a public key which is used to encrypt a message that can only be decrypted with a private key maintained by a user. The user's private key is kept in a browser's key store, protected with a passphrase or password. This system worked well for setting up a digital certificate with the Certificate Authority. The only snag at this stage was that the local computer was not keeping accurate time to within ten seconds of Greenwich Mean Time, which was a requirement for Grid use. Once the computer clock was corrected, the certificate system worked smoothly.

4.3.4 Executing jobs on the NGS

Test jobs were devised for trying the R system on the grid. The Rutherford Appleton laboratory Linux system node was chosen to try initially: *ngs.rl.ac.uk*. The first program was a simple R script to install and load a new R module called *affy* from Bioconductor, and return a list of the user R library modules available. It was necessary to set an environment variable to define the new R user library to the system. An early problem was not having 'write' access to the user R library.

Dr Hugh Shanahan from Royal Holloway College, London University, and Dr Farhat Memon who studied for her PhD at the University of Essex, contributed their R script for analyzing G-stacks from CEL files (an early form of the analysis described in section 3.6.1).

This script was used to verify the correct working of the system and took variable lengths of time, depending on how many CEL files were available in the experiment being analyzed. The R script was tried using an experiment with 352 CEL files which took 36 hours 2 minutes to run on a local machine.

Initially, the data in the form of CEL files had to be copied across to the remote machine, as did the R script. Here began a frustrating series of experiments to get the R script to run successfully. The common problem was **"User proxy expired"**! After seeking help from NGS Helpdesk staff, it transpired that the user certification only allows for a job to run for twelve hours, and then it is automatically cancelled and any files pertaining to it are deleted to tidy up!

The R script job was changed to analyze an experiment that had fewer CEL files, in this case 12 CEL files. When running this R script on local machines, it had taken between five and seven hours to run, depending on the machine. On the first NGS grid machine, the job had not finished in twelve hours. Information was sought concerning the running of jobs that take more than twelve hours. Instructions were received which concerned using a "renewal proxy". Having followed instructions carefully, an error message was received:- **"Error. MyProxyServer: wrong type caught for attribute"**. More advice was received and an error:- **"-save no longer supported"**. Apparently "- -save" should be used in Arguments. The following two attempts ended unsuccessfully after three days with **Status: Aborted, Reason: Job proxy is expired**. The next attempt aborted a week after being submitted with the logged reason: **"File not available. Cannot read JobWrapper output, both from Condor and from Maradona."**

Although long jobs were a challenge to the NGS Grid system, more smaller jobs were submitted with some success. This was followed with trials on other nodes of the grid which meant repeating the setup procedures to make programs and data available to each system. Table 4.1 shows the times taken for an R script to process the same six CEL files on four

Machine	Time taken	CPU and Memory
NGS RA node	3 hr 20 min	
Obelix	1 hr 14 min	Xeon 5140 2.33GHz 16Gb RAM
Asterix	1 hr 24 min	Xeon 5140 2.33GHz 16Gb RAM
bsstream3	1 hr 7 min	Xeon 5160 3.0GHz 8Gb RAM
s6320	0 hr 57 min	Xeon 5450 3.0GHz 32Gb RAM

Table 4.1: Table of varying execution times for the same job with 6 CEL files, which were run on the NGS Rutherton Appleford laboratory node and on four local machines

different local machines and on the one Rutherford Appleton laboratory node. The job run on a NGS grid node took over twice as long to run as it did on each of the local machines. Jobs were planned to run on other NGS nodes, but it proved difficult to get help when facing different problems on these other nodes, especially when a key systems administrator on the Helpdesk went on holiday.

The challenge of running the same R script with larger numbers of CEL files, like 300+ for example, was never overcome on the grid. These jobs proved to take too long for the user proxy limit of one week, which was an NGS standard. It is understandable that a job elapsed time limit be imposed so that all users get a fair share of the resources. However, eventually the conclusion was reached that the grid was not the best place to run jobs of the data intensive kind that are required in this study's bioinformatics research.

There could have been benefit in using parallel programming techniques for the analyses and thereby running larger numbers of shorter jobs, but since the majority of experiments did not have extremely large numbers of CEL files, it was not thought worthwhile to spend time on re-programming the R scripts.

4.4 The Future of NGS and NES

The NGS did not receive funding beyond 2010, but a national Grid service serving universities and research bodies has continued even though names have changed. It is recognised that computing needs vary as research progresses, and the flexibility of sharing resources

via a Grid has a part to play in ongoing research.

The GridPP continues to offer important computing resources to particle physicists, and the UK universities of Birmingham, Brunel, Durham, Imperial College, Liverpool, Manchester, Oxford, Queen Mary London, and Royal Holloway College are all involved in this collaboration. The Large Hadron Collider project at CERN is a user of GridPP, which contributes the equivalent of 40,000 PCs to the LHC's computing requirements for data analysis.

4.5 Conclusions on the use of NGS

It was found that for this research the NGS was not a suitable provider of computational resources. The main benefits of the NGS to a researcher have been listed in section 4.2.5. The main disadvantages of using the NGS grid were:-

- the lack of documentation about the systems and how to use them
- the fact that the NGS was not administered systems-wise fully as a grid
- its loads were uneven
- a user could run out of quota on a node
- the nodes were not always in synchronisation with each other
- service levels were unpredictable and patchy
- it was often difficult to get help from systems administration

The NGS was not a good provider of data storage services. It is not easy or speedy to make data available between nodes of a grid when a user has to do this for themselves. The systems administrators of the individual nodes had not been able to implement synchronization of data files for users. Therefore it was not considered viable to consider putting large

quantities of microarray data on the NGS for analysis. So the next chapter moves on to the consideration of and experience gained from specific computing clouds.

Chapter 5

Cloud Computing

The concepts of cloud computing have been introduced in chapter 2. In this chapter three examples of clouds are considered. First, an example of a public cloud is examined by considering the features of Amazon Web Services with its Amazon Elastic Compute Cloud and related storage and queuing services. It is an example of IAAS (see section 2.11.3.1). An example of the use of a local private cloud is described, with the results of a collaboration to investigate different queuing models within a self-service infrastructure container proposed for cloud implementation by Ibrahim Musa [61]. The third cloud computing example is of Windows Azure, which was a new cloud offering by Microsoft at the time that this research project was using it. Azure offered Web page access for cloud services before Amazon, and now offers hybrid cloud storage solutions to seamlessly extend a customer's own storage capacity for primary storage, backup, archive and disaster recovery.

5.1 Amazon Web Services (AWS)

Following the collapse of the dot-com bubble in the mid 2000s, Amazon reorganised its data centres on a **utility computing** basis and opened Amazon Web Services in 2006. Utility computing meant that Amazon was offering a **pay-on-use** service, similar to the model used by utility companies to supply water or electricity. Customers of AWS were able to avoid

the up-front costs of capital expenditure on powerful computers and air-conditioned rooms and buy computing facilities as a variable or operating cost on an as-required basis. The customer pays only for the resources actually used.

5.1.1 Types of Applications

AWS supplies the solutions to many types of computing needs, for example:-

- **Application Hosting**

  Companies can run their payroll, stock control, invoicing and many other types of application.

- **Backup and Storage**

  It is possible to store information and service backup solutions for vital data.

- **Web Hosting**

  Any dynamic web hosting needs can be satisfied with the scalable infrastructure platform.

- **Databases**

  A variety of types of database can be hosted, with the advantages of the security and scalability that the solution of cloud storage offers.

5.1.2 Amazon Elastic Compute Cloud (EC2)

The Amazon Elastic Compute Cloud is the web service which offers resizable computing capacity. One can configure the computer required (with its operating system) as a machine image, launch it as an ‘instance’, and control it as a machine from one’s home desktop. It is possible either to select a pre-configured machine image and launch it quite quickly, or to create a new Amazon Machine Image (AMI) containing one’s own applications, libraries,

and data. Figure 5.1 shows the flow of actions for a user to find or create and launch an AMI to use for running an application.

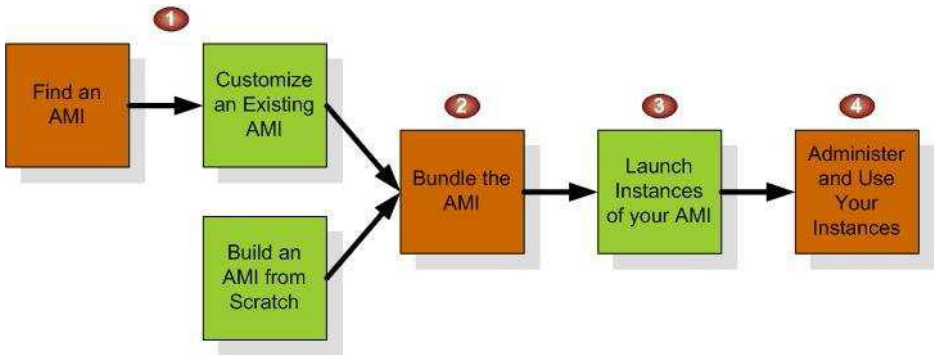


Figure 5.1: Flow within Amazon Elastic Compute Cloud (Source: Amazon documentation from *http://aws.amazon.com/*)

The initial experience of this research was in creating a machine image (AMI) which contained libraries for the scientific programming language R, and several packages built in R from the Bioconductor website. The resulting machine image was larger than it needed to be for the R script runs, and thus cost more to store on a permanent basis. It was quite slow to load and run, so a leaner system was created with only the essential requirements for R and Bioconductor. This achieved faster results and was cheaper to store and run.

Figure 5.2 shows the range of facilities and control mechanisms available to a user's application or indeed machine instance through Amazon's EC2 cloud service. The Elastic Block Store was used to access the public datasets of Ensembl Annotated Human Genome Data [62] and some of the NCBI datasets (details for access can be found at this website: *http://aws.amazon.com/public-data-sets/*). Amazon S3 (Amazon Simple Storage Services) was used for user data and for storing tailored machine instances. The Multiple Regions control was used to ensure that the machine instances were started in the same geographical region as where the Ensembl datasets were stored.

A description is given in our paper "Identifying the impact of G-Quadruplexes on Affymetrix 3' arrays using Cloud Computing" [57], of the use of Amazon Web Services to analyze public datasets. In this case it was shown that G-stacks adversely affect the

Figure 5.2: Some of the Amazon EC2 services available to an application or a machine instance (Source: *http://jayunit100.blogspot.co.uk/2013/02/making-your-ec2-instances-application.html*) [Accessed Sept. 30th, 2014]

reliable measurement of gene expression. The method used to demonstrate this finding was to use cloud computing to analyse 352 randomly chosen CEL files from 179 microarray experiments which used the human chip HG_U133A. The probes which had only a single G-stack of exactly four guanine bases and were mapping uniquely to an exon were filtered out to be used (for an explanation of unique mappings see section 3.4.2). A method of visualizing the correlation between pairs of these probes was devised. A contour plot was drawn to show the variation of the average correlation coefficient for pairs of probes containing G-stacks at different positions along the probe.

In Appendix A of a similar paper which concentrated on Affymetrix Exon arrays [63],

presented at the Conference of Integrative Bioinformatics 2009, the details are given of getting an application up and running on the Amazon cloud.

5.1.3 Experience of using Amazon web services

The details of setting up an Amazon account with a credit card, establishing security and authorization with public and private key pairs and so on is not discussed here as it has changed somewhat from the early days of AWS in 2008. The experience gained by this project in using Amazon Web Services and public datasets stored in Amazon Elastic Block Store was good, as long as running instances were closed down when experiments and analyses had finished. Otherwise the cost of using the services would continue to be charged without warning. In a similar way, the use of Elastic Block Storage for the project's programs, parameter data and results had to be well managed to minimize storage costs.

5.2 A Local Private Cloud

A collaboration with Ibrahim Musa came about when Dr Musa was looking for some real world biological data to use to test his private cloud which was the basis for his PhD study. He needed to find some data and analysis programs which would provide a range of challenges for his proposed self-service infrastructure container. This project was of interest because it was not merely a private cloud on which to test the R scripts which analyse microarrays, but it was also researching a new way of offering cloud services to scientists. It was agreed to share some of the microarray data and some R scripts to analyze the data. Dr Musa ran the analysis jobs, getting the same results, and using the differing amounts of data for each job to test out some job allocation algorithms within the private cloud structure.

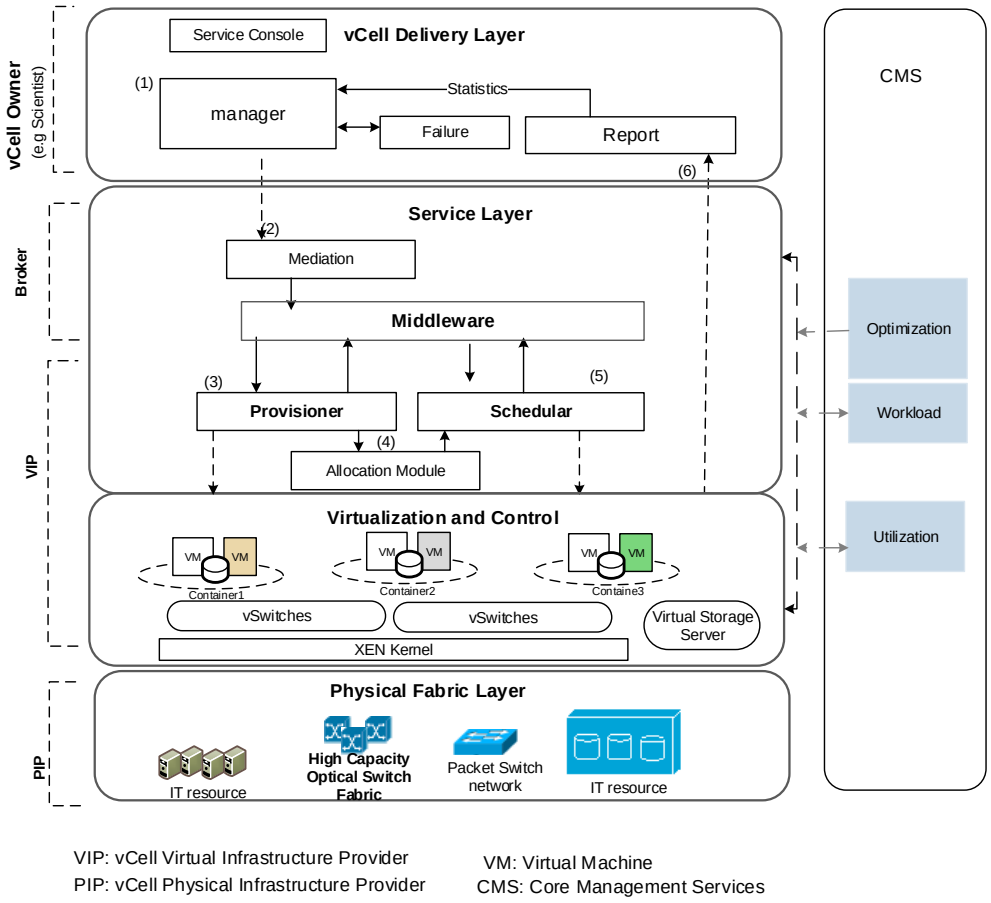


Figure 5.3: Framework and interaction of components in the proposed virtual infrastructure container *http://www.journalofcloudcomputing.com/content/3/1/5* [Accessed September 29th, 2014]

5.2.1 Description of the private cloud

Cloud Computing encompasses the service model where extremely large network and IT resources are virtualized and offered over the internet to increasing numbers of users. The users themselves do not have to understand the complex layers of machines and software which comprise the cloud, but are able to use the services at the application or machine level, depending on their needs. Dr Musa's PhD research was to create a virtual organization of resources in a holistic manner and with a single logical view which he termed a virtual service cell (vCell). Users would request one or more organised virtual machines rather than specifying detailed isolated resources. The network to connect the virtual machines, the

storage to support task execution requirements and the logic to manage all the components with minimum human intervention, would be provided with the machines. In this manner, user resources and management logic would be contained within the logical composition which simplifies the task of management and ensures high user service automation.

Figure 5.3 shows the implementation of Musa's virtual service cell. One can see the Physical Fabric Layer at the bottom, with the computers, network and switch components and storage servers, and the vCell Delivery Layer at the top which is where the scientist will interact with the cloud. The Core Management Services (CMS) are shown at the right hand side, interacting with the Service Layer and the Virtualization and Control layer in the middle. The actual computers that Musa used were ten four-year-old PCs with switching equipment and three storage servers. The details of the implementation of the private cloud can be found in Appendix D.

5.2.2 Significance of Musa's private cloud

Dr Musa's research was able to conceptualize and test an optimal framework for building a virtual container as a service (vCAAS). This model contributed a framework and the optimization techniques for next-generation cloud computing. The isolation of components within a vCell ensured that network and IT components operated with a vCell boundary which isolated external traffic and enhanced performance. This allowed service management to be attached to each vCell and to components within each vCell. The vCells within a vCAAS were each equipped with a privileged VM acting as the vCell manager (CM). The CM controlled communication between vCells and communication with external components outside the container (vCAAS). The per vCell management in vCAAS reduced complexities and simplified the technical and operational tasks of managing cloud infrastructure. In the vCAAS, a subset of network management functions was projected into a privileged virtual machine so that management tasks, such as utilization monitoring, usage accounting, failure recovery, adaptive control and scheduling tasks were managed on a per

vCell basis rather than on the whole datacentre infrastructure.

vCells were able to be replicated to simplify the initiation of new vCells for a new vCAAS container. In future development of this work it is envisaged that an automated vCell that encapsulates various service technologies would be used. This would enable a cloud administrator to provision a new tenant and configure the required services across the network in minutes. The vCell approach enables the efficient and automated scaling of hundreds of tenants and thousands of virtual machines with just a few steps and less complexity. The vCAAS approach ensures efficient delivery of applications, allowing any combination of network and IT resources to support the needs of the users' applications.

5.2.3 Experiments on the private cloud

The full description of the private cloud, its hardware, the infrastructure and middleware employed, the experimental setup details and full results and discussion are all given in the published paper: "Self-service Infrastructure Container for Data Intensive Application" [61]. A description of the experiments and figures of the results are given in Appendix D. Here the experiments and the key results are summarized.

The microarray data consisted of GSE experiments with differing numbers of CEL files and therefore differing amounts of data. Each experiment comprised one analysis job. The cloud was set up so that the vCAAS container was able to run a varying number of vCells, and in each vCell the analysis for one GSE experiment was run at a time. Three common job queue scheduling algorithms were used:-

- the Shortest Job First algorithm on K Queues (SJF-KQ)
- the Largest Job First algorithm on K Queues (LJF-KQ)
- the First Come First Served algorithm on K Queues (FCFS-KQ)

Initially the jobs were allocated randomly to the queues for each type of queueing algorithm. Then the quantity of data to be accessed for each job was assessed and used in

the provisioning management. Jobs were classified into groups according to the amount of data that each job required to download from storage. The VMs were given the appropriate bandwidth according to the amount of data required for that queue of jobs. Jobs which had more data to analyze took longer, and an estimated finish time was calculated and used to allocate jobs to one of three queues ($K = 3$). When the data for a job had been downloaded the VM's bandwidth was released to be used by another VM. This variation on the algorithms was called "with Lookup". Figure D.2 in Appendix D shows the improvement gained in thrashing values by the Lookup routines for all three job queue scheduling algorithms.

Using three queues Figure D.3 compares the versions of SJF task allocation algorithms which are called SJF-3Q and SJF-3Q-L (with Lookup) with FCFS-3Q-L to determine the elapsed time differences for each analysis job. The worst performing jobs (slowest) were those in the SJF without Lookup queues. These did not have the benefit of being scheduled according to the amount of data to download and thus suffered from the competition for bandwidth which produces thrashing. The First Come First Served with Lookup jobs outperformed those in the Shortest Job First with Lookup queues.

The concept of vCAAS was successfully implemented and this enabled experiments with algorithms for job allocation to be performed. Because the amounts of data varied for the GSE experiments, differences in data access times were exploited. The VM bandwidth for the data transfer was released after the data had been downloaded so that jobs could be classified into groups. This enabled the result that the holistic view of resources improved the performance of algorithms for resource provisioning and job allocation.

5.3 Windows Azure Cloud

Windows Azure is a cloud platform developed by Microsoft for running applications which can use computing and storage resources as needed. Initially the company had a need for an internal system which their own development teams could use, to manage multiple

machines, storage and networks in a disciplined, efficient manner. It was soon realised that this could provide a marketable platform for other companies to use and it eventually became Windows Azure. Microsoft data centres in Europe, North America and Asia have been set up to offer services across the internet through the Azure Cloud. It has all been developed since 2005.

Windows Azure varies from the Amazon Web Services in being originally a totally Windows based platform. Linux platforms were talked about, but not fully implemented until later. Services are routinely provided through Web pages, and developed using html web page language and Microsoft's .NET Framework interface libraries and services. Visual Studio is the integrated development environment provided for .NET software. The C# and Visual Basic languages are among those which can be used to develop applications in Visual Studio for use on the Windows Azure cloud.

The design and implementation of a Web Role to execute an R script application, and the use of Microsoft's Generic Worker are covered in chapter 6. This current chapter will introduce some basic features and design objectives of Azure. Figure 5.4 shows a simple view of the key elements of Azure that interact with a user's application.

5.3.1 The Windows Azure Programming Model

Windows Azure has been designed differently from the standard Windows Server type of operating system to improve life for applications in three distinct areas: administration, availability and scalability. The next sections examine how Azure works and how its advantages are achieved.

5.3.1.1 The Fabric Controller

Windows Azure runs in data centres containing many computers, and every Azure application runs on multiple machines simultaneously. The **fabric controller** is the Azure software which automates the control of all the computers in the data centres. It monitors the state

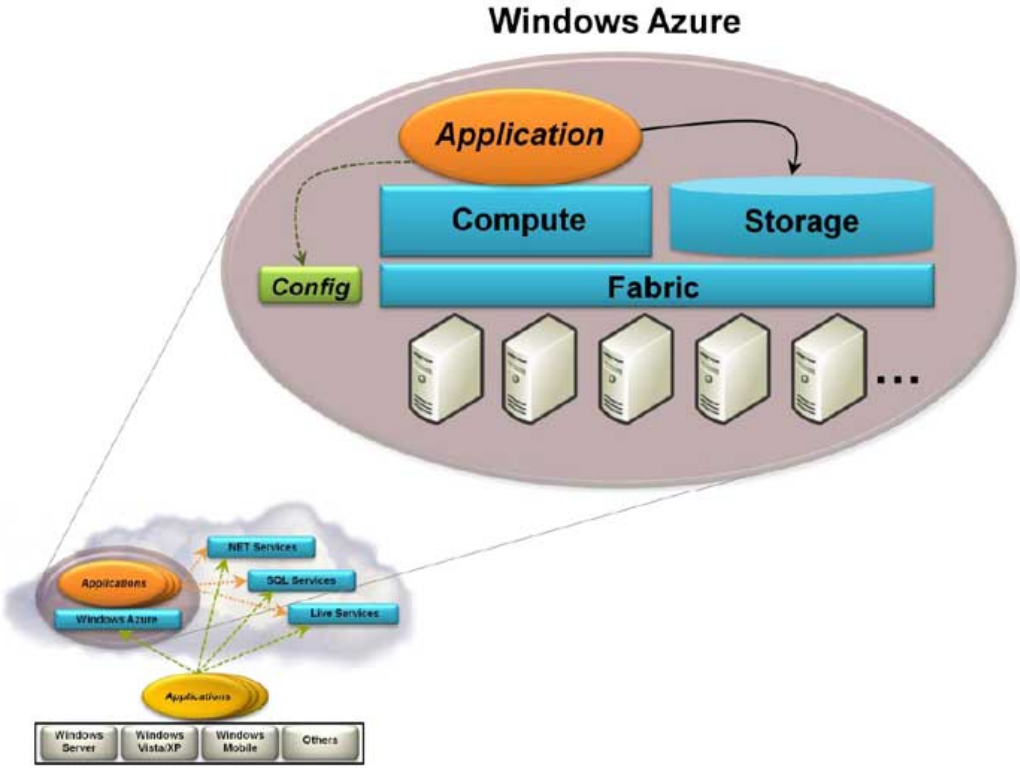


Figure 5.4: Elements of Azure (*http://solutions.devx.com/ms/developer-cloud/migrating/introducing-the-azure-services-platform-46874.html*) [Accessed October 14th, 2014]

of every node (or computer) and sets up what is needed according to a **service model**. It finds the right hardware and applies the right network settings. It monitors applications and hardware so that, in the event of a crash, an application can be restarted on either the same node or on a different node. The fabric controller also takes care of patching the operating systems that run on the nodes, doing so in a way which allows the services to continue running. It can automate the updating of various service software, including applications, in a way that allows services to continue running while updates take place.

5.3.1.2 Storage Services

Azure Storage provides services which are comparable to the file systems of an operating system. The environment of multiple systems in a cloud requires different storage solutions to deliver the advantages mentioned above of administration, availability and scalability. Azure uses Blobs, Queues and Tables as shown in Figure 5.5. It also supports a database

called **SQL Azure** which uses the same query language and interfaces as standard SQL databases.

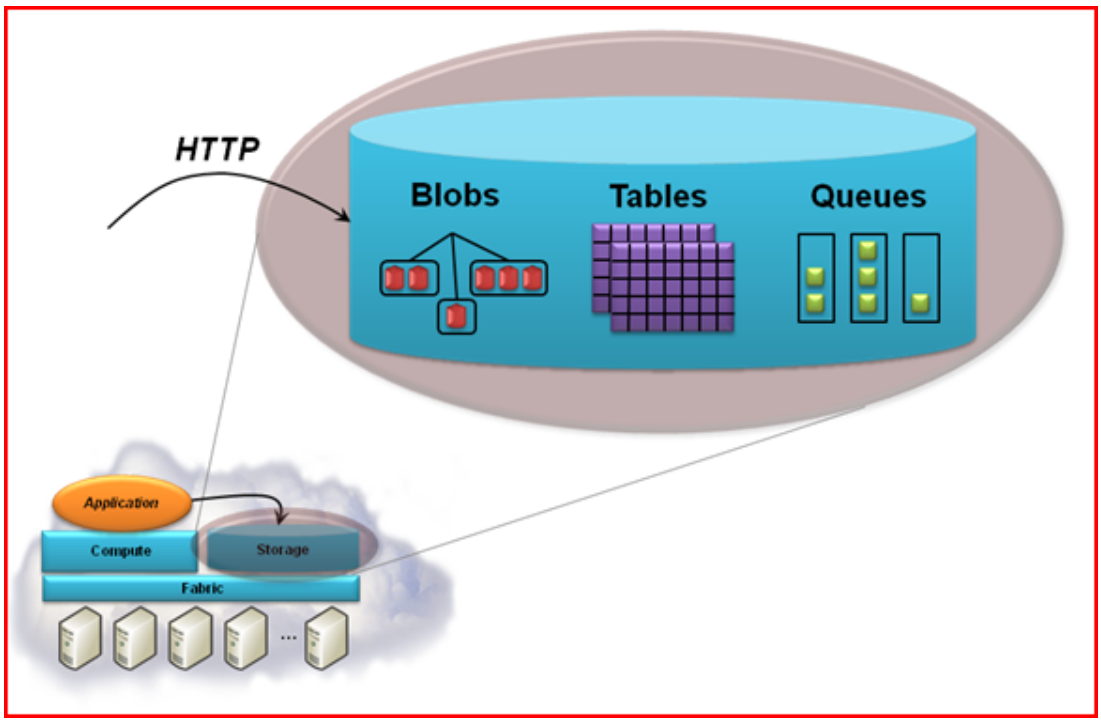


Figure 5.5: The three types of storage that Azure provides *(http://sijinjoseph.com/2008-/11/14/windows-azure-distilled/)* [Accessed October 14th, 2014]

- **Blobs (Binary Large OBjects)** are data files stored in containers which are rather like folders. Each blob and container has a unique URL, and the REST API (REpresentational State Transfer Application Programming Interface, developed in parallel with HTTP 1.1) can be used to create, delete, update and access blobs. Containers only have one level of data files in them and one cannot store containers within containers. Containers are used to set permissions and the policy for access control. If a container is set as publicly accessible, then any blobs in that container can be accessed by anyone over the public Internet, using the usual *http://* protocol.

- **Queues** provide for storage and delivery of messages to an application. The queues themselves must be resilient so that no messages are lost. Azure guarantees that any number of messages can be queued, that none will be lost and that a receiver will

always be able to dequeue them. In providing cloud services, queues are used to help decouple various components of the total sytem like backend servers, frontend servers and storage servers so that if a component fails, the failure does not cause the whole system to crash. The lengths of various queues are used to scale resources, for example, adding worker nodes when queues build up beyond a pre-arranged length and standing down worker nodes when the queues are small.

- **Table storage** enables massive amounts of structured data to be stored cheaply and efficiently. Like relational databases, the data in tables is stored in the form of *entities*, each with a set of *properties*. Developers of applications have control over how this data is partitioned physically. The table storage is not queried with SQL, but with an ADO.NET Data Services REST interface which is not quite as flexible as SQL. Table storage is optimized for cheap and highly scalable storage.

- **SQL Azure** is the way that Azure has chosen to achieve full database flexibility in the cloud. The databases are stored in Microsoft's data centres, and accessed using normal SQL tools and code. The price of SQL database storage per GB in Azure is several times that of the Table storage because of, for example, the overheads in making detailed fields and relationships within the data easily available.

All types of data in Azure storage is stored with multiple copies which are managed by Azure itself, so the user does not need to plan their own backup services. Several processes are in place to ensure that data is not lost.

5.3.2 Application code: web roles and worker roles

A **Web role** is largely intended for logic that interacts with the outside world via HTTP. The code can be written in Java, C#, VS Basic or other languages, and can interact with IIS (Internet Information Services), ASP.NET, WCF (Windows Communication Foundation), PHP and so on.

A **Worker role** is provided for code that interacts with the outside world in various ways, not limited to HTTP. For example, a worker role might contain application code that converts videos into a standard format or calculates the risk of an investment portfolio or performs some kind of data analysis. It would typically be triggered by an entry in a table, or a message in a queue.

A typical application might use a web role to accept HTTP requests from users, then hand off the detailed work required to one or more worker roles. It would pass the work on by putting a message into a job queue designed to start the applicable worker role. Parameters could be passed in the message. The primary reason for this breakdown into two roles is that dividing tasks in this way can make an application easier to scale. A developer needs to create a **service definition file** that names and describes the application's roles. This file also specifies other information used by Azure to build the correct environment for running the application. The **service configuration file** also needs to be created. It specifies the number of role instances to deploy for each role in the service, the values of any configuration settings, and the thumbprints for any certificates associated with a role.

5.4 Summary of Clouds, their features and relevance to bioinformatics research

Three examples of cloud computing solutions have been described in this chapter. They are all relevant and useful for research in bioinformatics and have each been tried by this research project. They can each satisfy the requirement to run R scripts and store large amounts of data for easy retrieval and analysis.

The public cloud Amazon Web Services had a head start by being early in the field to offer utility computing on a wide scale. It also proved useful by hosting a significant quantity of biological research data with the public datasets of Ensembl Annotated Human Genome Data [62] and some of the NCBI datasets. These were used to research the effects

of G-stacks and G-quadruplexes on the measurement of gene expression. This work was published in our paper "Identifying the impact of G-Quadruplexes on Affymetrix $3'$ Arrays using Cloud Computing" with lead author Dr Farhat Memon [57].

The private cloud developed by Dr Musa at the University of Essex has shown how small cloud computing solutions can benefit bioinformatics research. As a computer scientist, Dr Musa was interested in furthering research into next generation cloud computing solutions. He used the data and R scripts supplied by this research project to demonstrate how R scripts can be run to process different sized datasets, with the objective of improving queueing times and minimizing the total time taken to run many jobs. The concept of vCAAS was demonstrated as providing a holistic view of resources which could improve the performance of algorithms for resource provisioning and job allocation.

The public cloud Windows Azure was introduced later on the scene of utility computing offerings and is still much smaller in business terms than AWS. Microsoft say that they had developed a cloud for their own internal work, and then decided to offer it commercially. From the outset it was different from Amazon in being Windows-centric for application development and friendly to Web developers because of its .NET Framework, Visual Studio and Windows programming language offerings such as Visual Basic, C# and Visual C++. On the other hand, Azure can suffer because of Microsoft's occasional non-standard quirks such as experienced by web developers trying to interface with Internet Explorer.

Windows Azure was used for the bulk of the research analysis for this work, and many early problems had to be overcome as Azure was in its infancy and undergoing changes at the time. Much persistance was required and the next chapter will explain the details of the data uploads and job submission procedures which were utilized to perform the first wide scale analysis of G-stacks.

The advantage of clouds being able to spread computation across many VMs concurrently was not exploited in this research study. The type of bioinformatics research analysis jobs chosen were mostly not long-running. Timings for certain jobs are shown in the next

chapter (see Figure 6.7 where the longest jobs took about five minutes). It was therefore not appropriate to employ parallel programming techniques within the programming routines used in this study.

Chapter 6

The Azure Cloud

Windows Azure is a cloud service that has been introduced briefly in section 5.3. Here the cloud services of Azure will be discussed in more detail, together with experiences gained through using Azure to analyze GeneChip data. The application was developed from work done by Shanahan *et. al.* [38] to analyze the effect of G-stacks in the probes of the Human HG_U133A Affymetrix GeneChip. They showed that G-stacks exhibited a level of hybridization that is unrelated to the expression levels of the mRNA that the probe sets are intended to measure. Further details of this analysis are given in section 3.6.1. A similar analysis was performed in this work on the Azure cloud, with follow up work to estimate similar effects in other GeneChips and other species.

This chapter will concentrate on the experience of using the Azure cloud: developing the R scripts for the application, developing a wrapper called GWydiR for submitting jobs to Azure, and loading the public microarray datasets into Azure storage. Future chapters will explain the results that were found from analyzing the effect of G-stacks in the chip probes.

6.1 Azure Services

Azure is the name given to PaaS cloud platform services from Microsoft. Online Services is the name given to SaaS cloud platform services from Microsoft. A **service** hosted in Windows Azure consists of one or more **web roles** and/or **worker roles**. A **web role** is an ASP.NET web application (more recently called AzureAppFabric) which is accessible via an HTTP or HTTPS endpoint (webpage) and is commonly the **front end** for an application. **Worker roles** are the background processing applications which typically access data and perform the required computation.

Azure services may use one or both types of role, and can run multiple instances of each type. **Role instances** can be added or removed based on demand and they allow applications to quickly and economically scale up or down when the need arises. The **fabric controller** is aware of every Windows Azure application and performs the scaling, load balancing memory management and reliability functions required across the cloud services.

Azure **storage services**, described in section 5.3.1.2, consist of blobs, tables and queues. SQLAzure provides database services via a version of the SQL Server.

6.2 Initial Azure applications

The early implementation of any Azure application required the installation of the following software:-

- MS Visual Studio 2010
- Windows Azure Tools
- Windows Azure SDK (Software Development Kit)
- Windows Server IIS
- Windows Azure Platform Management Tool

- Windows Azure Platform Training Kit

With the first four of these software tools as a minimum, an application can be generated and developed to run in the cloud via a webpage. The last two software items were required to make the Azure demonstrations work.

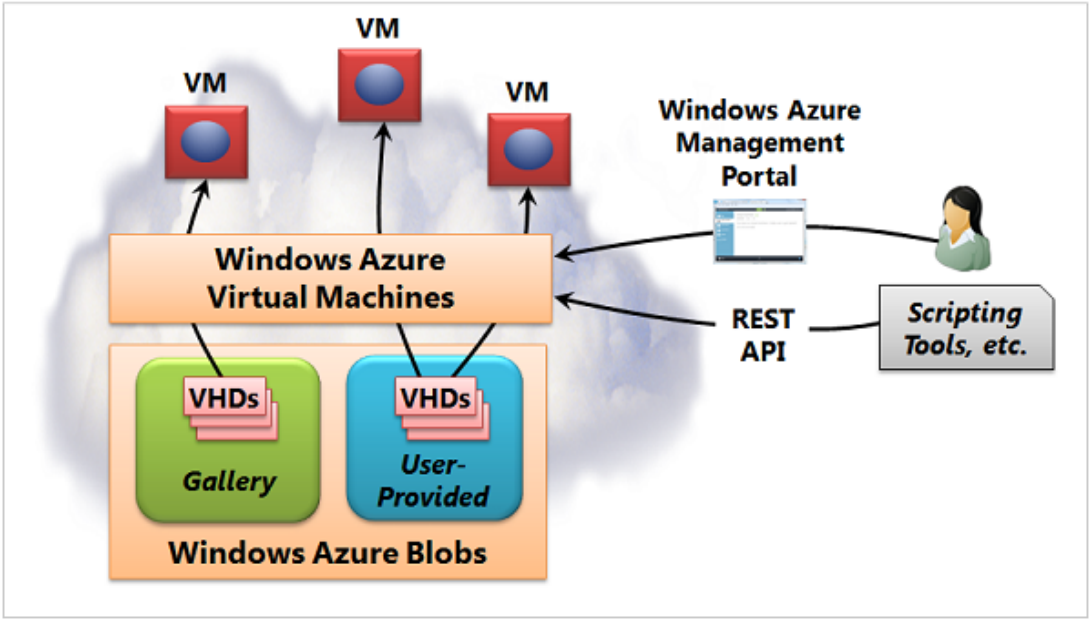


Figure 6.1: Simple flow representation of Windows Azure *(http://sijinjoseph.com/2008-/11/14/windows-azure-distilled/)* [Accessed October 14th, 2014]

The first application developed for Azure was a simple "Hello World" application which launched a webpage with a message. This was relatively straightforward to achieve after registering with Windows Azure. The second application was a short R script using a web role, which took a little longer to get working. The third application was an R script with data sent from the local PC. Figure 6.1 shows both the Management Portal and the REST APIs as ways to launch and control applications in the Azure cloud. These first applications were all developed via APIs (application program interfaces). The Management Portal was not available when the work was first done.

It was possible to run a cloud application in a 'Compute Emulator' while testing it. This emulator launched a browser to show a running instance of the application locally. With

this approach the debugger in MS Visual Studio could be used to step through the code and check the data and the processing at all stages. Figure 6.2 shows a screenshot of an early webpage used to launch an R script job. A hosted service was created and deployed from the Azure Management Portal. This took several minutes to get to the "Ready" stage. Then the application was tested from any device which could serve an internet webpage. It was important that the correct Storage Account was referenced in the C# programs in Visual Studio. Each Storage Account had to be managed with the correct Connection Strings.

It was possible to create as many different service configurations as required and keep them for different uses, such as local debugging, using different storage areas, cloud testing and so on. There was also the issue of adding a **certificate** to a role when running it in the cloud. A certificate in this context is a small file which holds previleged information called a **key** and carries a password for authentication c.f. the keys and authority needed to access services on the NGS grid, as in subsections 4.3.2 and 4.3.3. The properties of the web role had to be set appropriately with any development storage areas required having their addresses set correctly.

An application was developed using R to allow the user to specify a file to be uploaded to the cloud where the R script was run with data files supplied to it and results returned. The finished jobs were shown on the webpage below the submission section, see Figure 6.2. This application was developed and expanded to allow several datasets to be processed by the same R script analysis program with corresponding results obtained. The data file names were passed as a list in an Excel data file.

6.3 Generic Worker within Azure

The Generic Worker (GW) is a special type of web role that has been developed by Microsoft within the framework of the Venus-C project (see Appendix E for details) to provide a standard web role for launching, running and managing applications and their files.

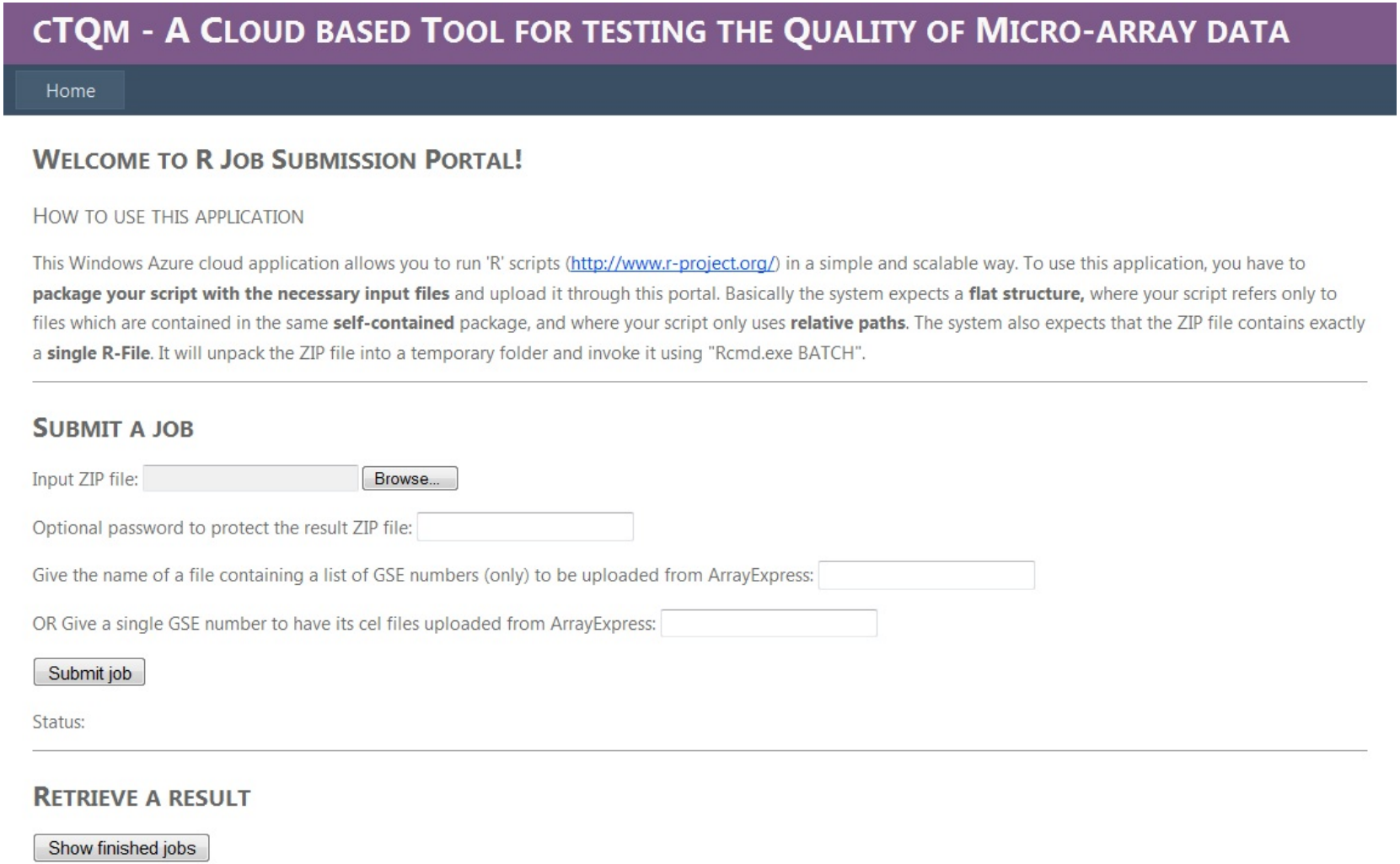


Figure 6.2: Screenshot of webpage used to launch an R script

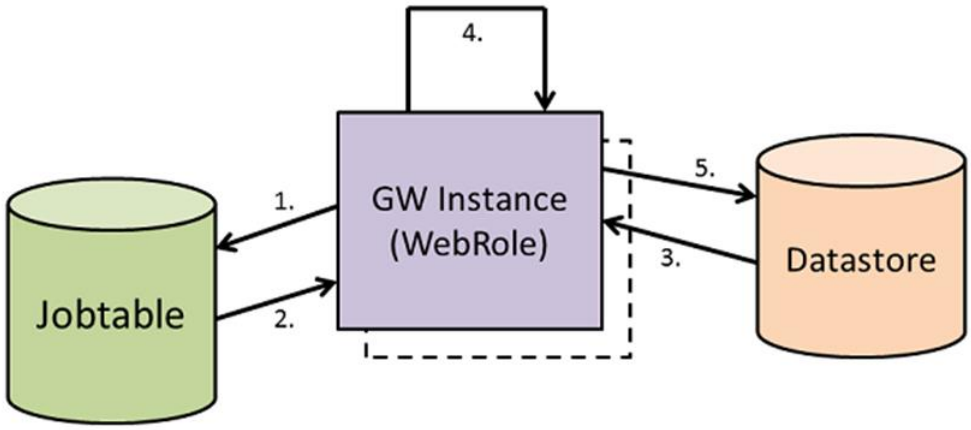


Figure 6.3: Functionality of the Generic Worker (GW) (Source: Early Venus-C documentation)

The GW libraries can be accessed from *http://resources.venus-c.eu*. GW was clearly applicable to the requirement for running analysis jobs repeatedly on different sets of microarray data. The numbers on Figure 6.3 refer to the following actions taken by the Generic Worker instances:-

1. Looking for submitted jobs
2. Fetching job details
3. Download application and requred files
4. Run application
5. Write back result files

Figure 6.4 shows in more detail the architecture of how the Generic Worker was designed to interact with the client user, the storage devices, the accounting routines, job queues and status table within the Azure cloud. The GW libraries can be used within a C# program to submit and efficiently manage jobs submitted to a set of worker roles, as illustrated in Figure 6.5. In this framework the software can run in two modes. In the first mode an application is uploaded to Azure storage together with a description of the application (in

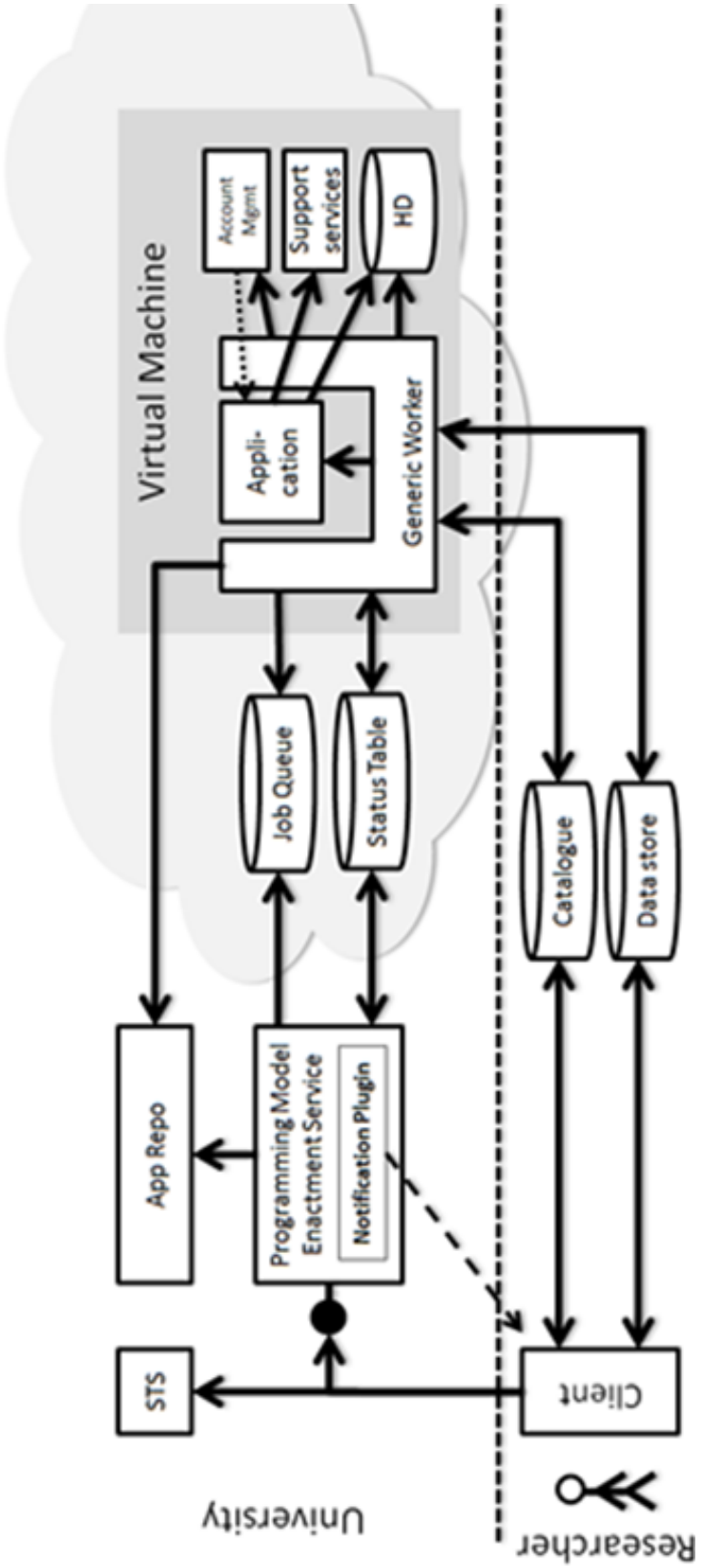


Figure 6.4: Design flow of Generic Worker web role (Source: Early Venus-C documentation)

particular the expected list of parameters that will be used in the application). In the second mode the application is transferred from Azure storage and launched with a specific set of parameters. The GW provides efficient job management and allows a means of scaling a task. Activation or termination of instances of worker roles can be called via the application code or from within a Windows power shell script. Additional software can be uploaded and installed if required at the start of each run.

6.4 Development of GWydiR for job submission

GWydiR was the name given to the wrapper written in R to streamline the process of submitting R jobs to the Generic Worker in Azure. GWydiR asks for a series of parameters and file names which it passes in appropriate ways to Azure for starting web roles in virtual machines. The parameters included:-

- **doUpload y/n** To specify whether the application data needed to be uploaded afresh or not
- **dataKeyFile** <**path/filename**> Key file for the data container storage
- **RFileName** <**path/filename**> The R script that will be run
- **userZipFileName** <**path/filename**> A zip file of all the extra files needed to be installed in the working directory to run the R script
- **csvFileName** <**path/filename**> A csv file of all the datasets to be analyzed
- **outputRoot rgw** Root for the log files
- **appKeyFile** <**path/filename**> Key file of the application's container storage
- **myApplicationName rgwtest** The application name
- **serviceURL** URL for webpage to inspect progress

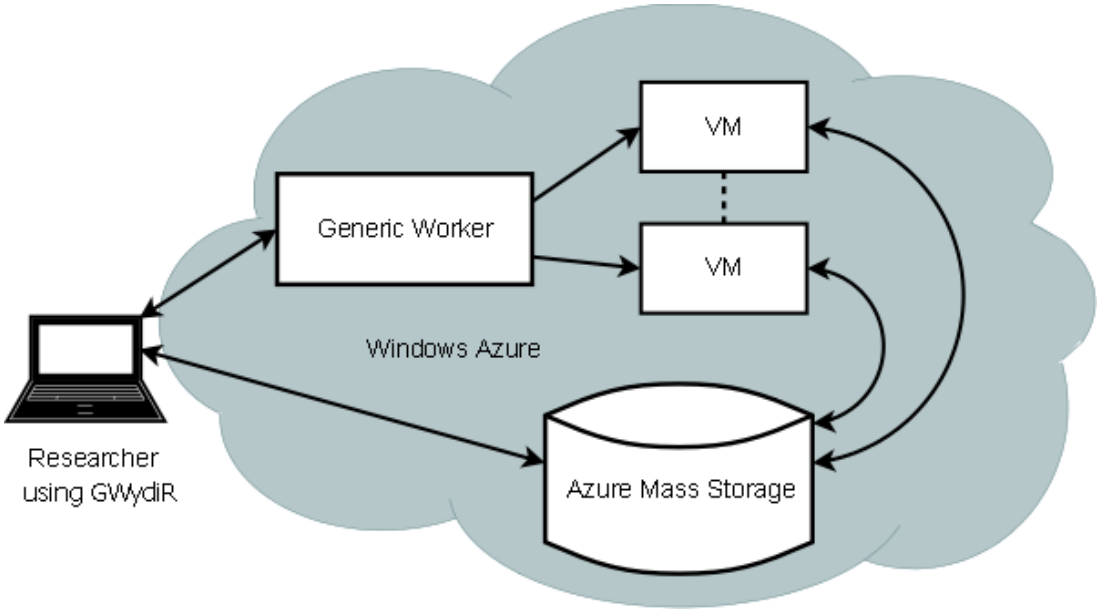


Figure 6.5: The use of GWydiR with Azure

Figure 6.5 shows the Generic Worker receiving information from a researcher to set up VMs which run web roles (in this case) to run R scripts and process data from Azure storage blobs. The source code for GWydiR can be downloaded from:- *http://gene.cs.rhul.ac.uk/RAzure/GWydiR.zip*. Full details on setting up an Azure account, running GWydiR and a sample R script on Azure is given in the Supplementary Information section of the paper "Bioinformatics on the cloud computing platform Azure" [64]. The paper gives a full discussion of cloud computing solutions for bioinformatics and how Azure fits in to cloud offerings.

Once the GW and GWydiR software had been set up and tested, it was a straightforward matter to scale up the number of VMs to run the analysis jobs. Each experiment was submitted as a separate job. Earlier in the use of Azure, lists of experiments had been submitted for analysis runs. It was found that the list approach was less easy to control and scale because sometimes an experiment within the list would fail, for example through a shortage of disk space. By starting each experiment as a new job with fresh disk space this problem was minimised.

6.5 Using Azure storage

The first thing to mention is that Azure Storage is distributed widely, both geographically around several areas in the world, and over many machines. The user has the choice over where in the world their data is stored, and it makes sense to choose a geographic location close to where the processing of the data will take place.

The four key data services offered by Windows Azure: blobs, queues, tables and a database have all been described in section 5.3.1.2.

The storage is managed with various distributed software techniques to make sure that it remains available and reliable if any failures should occur in machine or network hardware elements. The user is told that "all data is replicated multiple times" [65] page 130, and that recovery from hardware failure or data corruption in any of the replicas, "happens under the covers".

6.5.1 Preparing and accessing Windows Azure storage

Initially a Storage Account has to be applied for and created. The price for storing 100GB of data for one month in March 2013 was $US 9.50 for Blob storage, and $US 175.83 for SQL database storage. Each storage account needs a unique name, an optional description and some other information. One also has to choose a geographical location, such as North Europe, or Eastern USA. It is possible to store data and application program code in *Affinity Groups*, to keep them near to each other geographically.

6.6 Security and Accounting

Microsoft was keen that Venus-C projects should test and use all the security and accounting features which were provided with Windows Azure. However, unfortunately there was not time in the life of the project to incorporate these additional features into GWydiR and the applications, so the minimum of security was used and none of the Azure accounting.

6.7 Loading data into Azure storage

The expression data from the Human chipset HG_U133A was chosen as the first array data to be uploaded to Azure storage as it was a chip which achieved wide use and popularity as a reliable GeneChip for human tissue samples. It was used by Shanahan *et al.* in their paper on the bias of G-quadruplex formation [38].

In this initial work on Azure, 576 GSE experiments deposited before May 2012, consisting of between 2 and 200 CEL files were downloaded from ArrayExpress using packages provided for use with the R statistical programming language. R scripts were used to read a list of GSEs from a data file and download all CEL files for each GSE. The GSEs were then uploaded to Azure storage using a Web role program written in C# and ASP.NET, as described earlier in this chapter.

6.7.1 Issues found during uploading

The following issues had to be addressed during the time of uploading GSE data files from the EBI public database ArrayExpress to Azure Mass Storage blobs:-

1. **GSEs with only 1 CEL file**

   GSEs with only 1 CEL file were not uploaded as they do not contain enough data to determine significance in the analyses.

2. **GSEs with no CEL data**

   It was found sometimes that even when experiments were reported to have raw data in the form of CEL files available on GEO, the raw data files were not available on ArrayExpress. This condition did not stop the list of GSE experiments from uploading, but resulted in empty blob containers being created which subsequently had to be removed individually by hand.

3. **It appears that a CEL file is corrupted**

Occasionally the *affybatch* package from Bioconductor gives a message for example, like *It appears that the file GSM318883.CEL is corrupted.* A CEL file like this was deleted from its GSE as it was not known how to correct the corruption. When this message occurred, it happened consistently for that particular GSE and CEL file.

4. **A CEL file may be truncated**

   Sometimes a CEL file had to be deleted because of a message for example, like *Warning: found an incomplete line where not expected in GSM267392.CEL. The CEL file may be truncated. Successfully read to cel intensity 1343267 of 1354896 expected.* This happened consistently for a given GSE and CEL file.

5. **Phenodata or FeatureData may be missing**

   Processing was stopped by messages for example, like *Error in download.file(samples, sdrffile, mode = "wb", quiet = TRUE, cacheOK = FALSE) : cannot open URL 'http://www.ebi.ac.uk/microarray-as/ae/files/E-GEOD-8859/E-GEOD-8859.sdrf.txt' does not exist or is empty. The object will not have featureData or phenoData.* FeatureData and PhenoData included for each CEL file the metadata relevant to that biological sample. It might describe a type of tumour, a symptom, a drug treatment applied or several factors relevant to the sample. This data would be important for analysing fold change information. In some cases the file names varied slightly from the standard, which meant that automatic downloads did not find them.

6. **The request to ArrayExpress hung up**

   Occasionally it was found that there were CEL files there, which were downloaded, but could not be uploaded because the script hung up before normal completion. The reason was not easy to determine, so, as this only happened a few times, these were dealt with individually.

6.8 Running analyses on the Azure cloud

6.8.1 Validation of the service

Given the new architecture employed, it was necessary to test routines and methods by repeating an analysis performed by Shanahan *et al.* in their paper on the biases in expression data caused by G-quadruplex formation [38]. For this it was important to use only the 176 GSEs from the HG_U133A GeneChip which were available to them when they performed their analyses.

The initial test chosen was the estimation of the extent of bias of G-stack probes (see section 3.5) among other HG_U133A data. Figure 6 in the paper by Shanahan *et al.* [38], shows a chart of the average correlation between G-stack and C-stack probes for 176 HG_U133A GeneChip data sets deposited at GEO. These average correlations were recalculated to validate the use of R scripts and GEO data on the Azure Cloud, as results could be compared with those originally obtained. The chart obtained by this work on Azure is shown in Figure 3.10 in chapter 3 where the method was described. The charts are similar which gives confidence of reproducibility and validates the R scripts being used.

6.8.2 Timing of jobs using Azure and using local machines

The one-time uploading of data into Azure mass storage has been discussed. Attention is now turned to considering the use of cloud computing compared to the use of a local machine for analysis computations. It is important to understand how the run times on Azure compare with a locally run calculation. Date and time stamps were inserted into the R scripts to be able to compare the elapsed time taken to process the microarray datasets on cloud and local computers. It was recognised that there might be significant time taken to load the datasets from Azure mass storage to the R working storage in the VM where they could be used in computations, so the data and time stamps were written to a log at the beginning and end of loading each set of CEL files. This load time is generally negligible

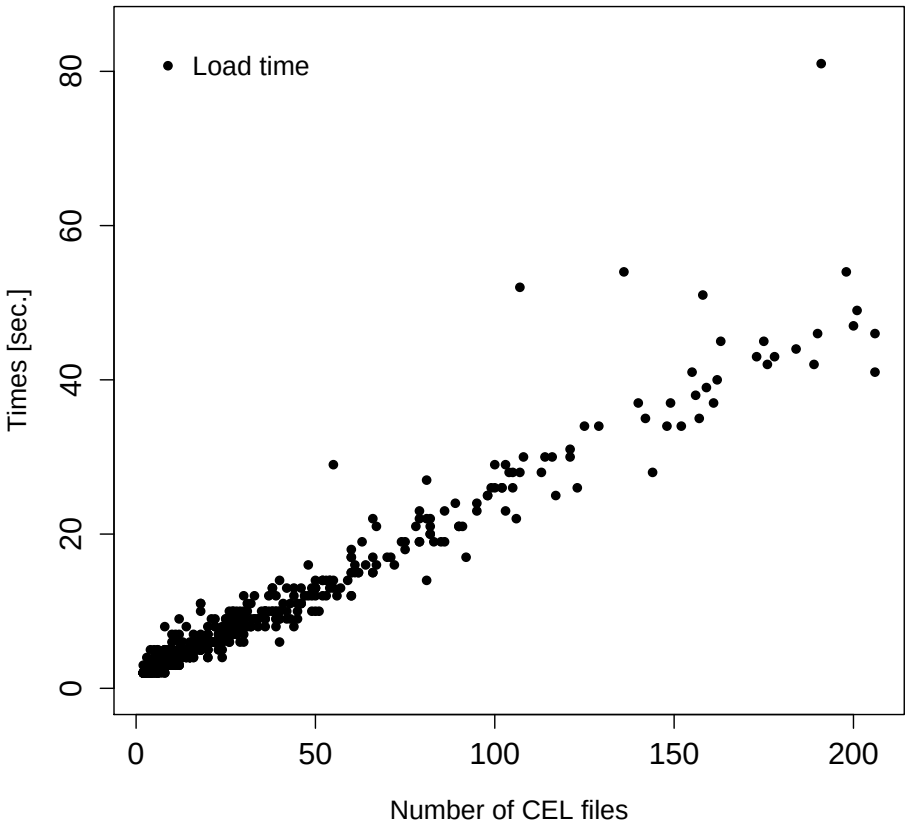


Figure 6.6: Time taken to load microarray data from Azure mass storage to R working storage. Plot shows the time in seconds taken to load each of 576 datasets from Azure blob storage to local VM disk space, in terms of the number of CEL files in each GSE experiment.

in the local setting. Similar date and time stamp output was also written to the log at the beginning and end of performing the main normalization and G-stack comparison analyses. Each of these elements will be considered.

6.8.2.1 Load time

Figure 6.6 shows the timings for loading data files from Azure blob storage to R working storage. All 576 available HG_U133A experiments are included. The elapsed time for loading an experiment comprising two CEL files (about 23 MBytes for text CEL files) was typically about two seconds, and for an experiment comprising 200 CEL files (about 2.25

GBytes for text CEL files) was around 45 seconds. The reasons for varying sized CEL files have already been outlined in section 3.2.6.

The outlier experiments in Figure 6.6 depend on whether the CEL files were stored in binary format (shorter load times than the trend) or character format (longer load times than the trend). The largest outlier, having a load time of about 80 seconds, had a combination of binary and text CEL files.

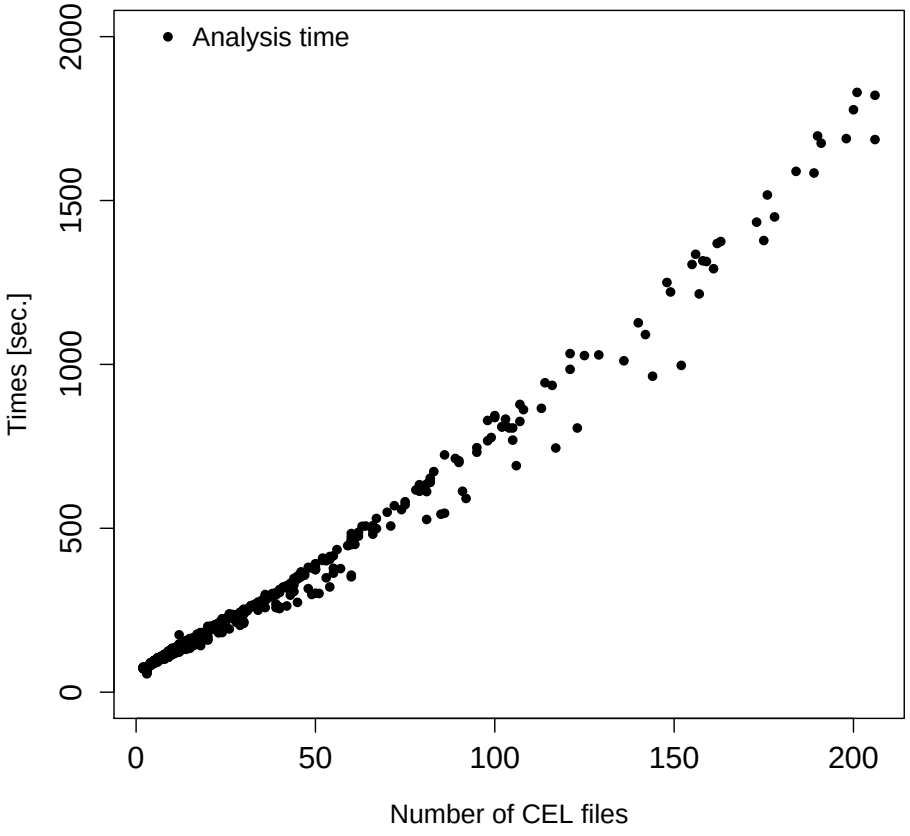


Figure 6.7: Time taken to analyze data with R script using Azure. Plot shows the time in seconds taken to analyze each of 576 datasets, in terms of the number of CEL files in each GSE experiment.

6.8.2.2 Run time on Azure

Figure 6.7 shows the timings of the 576 analysis runs (i.e. how long the R scripts ran on an individual VM once the data had been loaded). The outliers below the trend of the dots

are experiments with binary format CEL files, which are a little quicker to process than the character format ones.

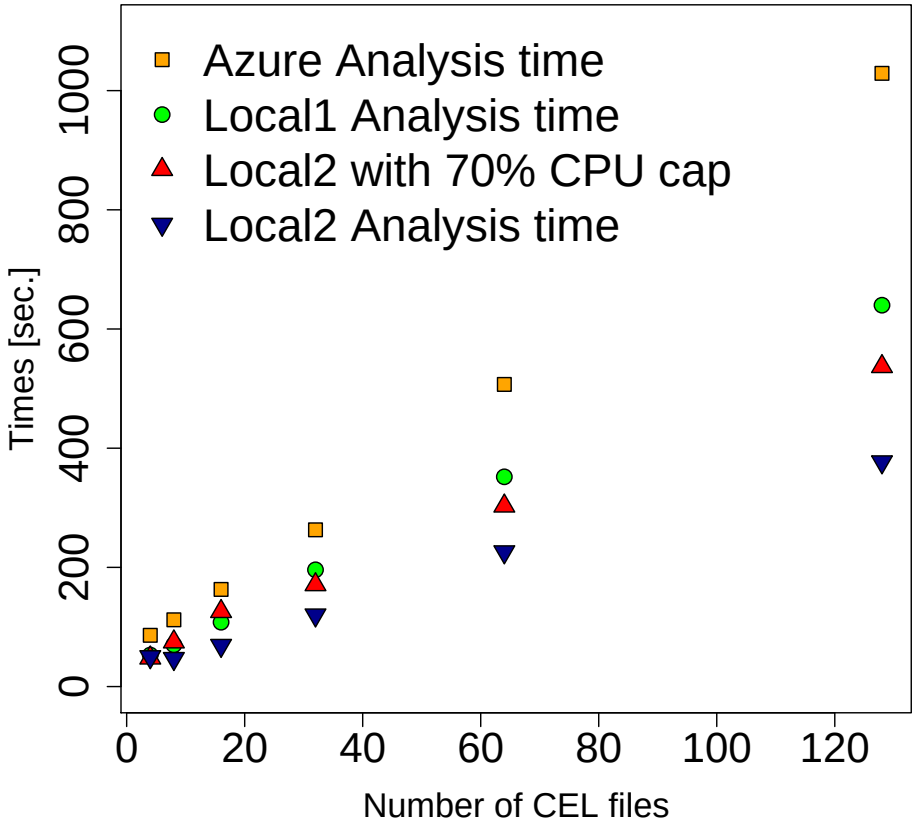


Figure 6.8: Comparison of analysis times between cloud and two local machines. Plot shows the time in seconds taken to analyze each of six particular experiments, in terms of the number of CEL files in each GSE experiment. The particular experiments were chosen because they had 4, 8, 16, 32, 64 and 128 CEL files, to give a range of experiment data amounts. The machine labelled Local1 had a CPU clock speed of 2.13 GHz, and the machine labelled Local2 had a CPU clock speed of 2.24 GHz. The 70% CPU cap was added to the Local2 machine to crudely estimate the slower 1.60 GHz stated clock speed of the Azure VM.

6.8.2.3 Comparison of elapsed times for analysis routine between Azure cloud and two local machines

A representative set of six experiments was chosen to repeat the analyses which had been done on the Azure cloud, on local computers. The experiments had a range of numbers of

CEL files to ensure that they were representative of different lengths of jobs. Figure 6.8 shows the results and the details of the clock speeds of the different local computers. The Azure cloud VMs all had a stated clock speed of 1.60 GHz. It is difficult to reproduce exactly the configuration of an Azure VM and therefore a variety of different local computers were used for comparison purposes. Local1 is a standard Windows 7 machine with a 2.13 GHz processor. Local2 has a 2.24 GHz processor that runs Windows as a virtual machine. As Local2 is run as a virtual machine it is also run with a 70% execution cap to crudely reproduce the nominal Azure VM processor clock speed.

The results in Figure 6.8 show that in all cases the Azure VMs run more slowly than the local machines, taking roughly a factor of two times as long as the middle local case.

6.9 Summary of experience of using the Azure cloud

For a researcher accustomed to using Linux systems, the structure and classes employed in developing Windows-centric applications for Azure was a steep learning curve. It was helpful to have the demonstration applications for creating a first Web role and Worker role. Visual Studio was very helpful in developing a simple working application using the C# language. Microsoft gave some assistance in providing the interface required to upload and run the first R script. In essence the required R package with its libraries had to be zipped up into one file and uploaded to Azure storage. This zip file was called and unzipped from within the C# Web role. If the required level of R changed, or if addional R libraries or CDF files were required, then this step of uploading a new zip file had to be performed.

The majority of bioinformaticians who use cloud computing have made use of Amazon's EC2. This fact is stated by Shanahan *et al.* [64] who attribute it to the provision of a stable software stack with an associated large community of users who can provide support and solutions specific to a researcher's domain. Nonetheless, it is clear that Amazon has viable competitors in the marketplace of cloud-based solutions, and that other offerings should be

considered. The Azure cloud offers the PaaS infrastructure which allows users to develop tailored interfaces for specific executables that run in a batch mode. Despite the learning curve for researchers who do not have a background in developing C# code, the tailored interfaces can be created with a little effort requiring approximately 400 lines of code, much of which can be gathered from templates. The initial configuration to run R scripts was complex but the access to Azure services in general has been streamlined and simplified in recent months by Azure developers.

The pricing of Azure is comparable with other clouds though this is highly dynamic. It is a competitive marketplace and volumes are important to the big players like Amazon. It has been shown that results generated locally are reproduced by equivalent Azure runs. It is suggested that Azure is slower for run times by a factor of 2 or more than local machines, though care must be taken as it was not possible to make a like-for-like comparison using precisely the same CPU type, memory and exact version of the Windows operating system.

A big factor in the choice of cloud computing service is whether a particular cloud already hosts the public databases that a researcher wishes to analyze. It is not really viable for an individual research project to upload vast amounts of data to a public cloud in order to carry out wide scale analyses, because of the time the uploading takes and because the cost of storing the data is charged by the GByte per month. It is noted that Amazon already provides a number of relevant datsets free of charge to users of their cloud services, such as the data from the modMine project which gives flexible access to the modENCODE data [66].

Chapter 7

The Analysis of Human GeneChip Data

7.1 Introduction

This chapter explains the use of the three analysis methods described in section 3.6 on more of the available human HG_U133A microarray data to determine the extent of G-stack bias in the experiments. This chapter also examines the effects of other variables in the experiments. The Azure cloud was used to store the datasets and to process R scripts for the analysis work.

In section 3.6.1 it has been shown how an analysis of the average correlation of expression levels over all CEL files using a sample of 1000 G-stack probes can be used to estimate the bias introduced by G-stack probes into microarray experiment data. The example used was from the GeneChip HG_U133A of *H. sapiens*. The average correlation of expression levels of C-stack probes was used as a control, and Figure 3.10 shows a bar chart using data from 176 experiments. The average correlation for the C-stack probes is much closer to zero than for the G-stack probes. These results are similar to those obtained by Shanahan *et al* [38], so that the method and code in the data analysis R scripts have been validated. This chapter describes more detailed results confirming the bias that G-stacks introduce into microarray data.

7.2 Method

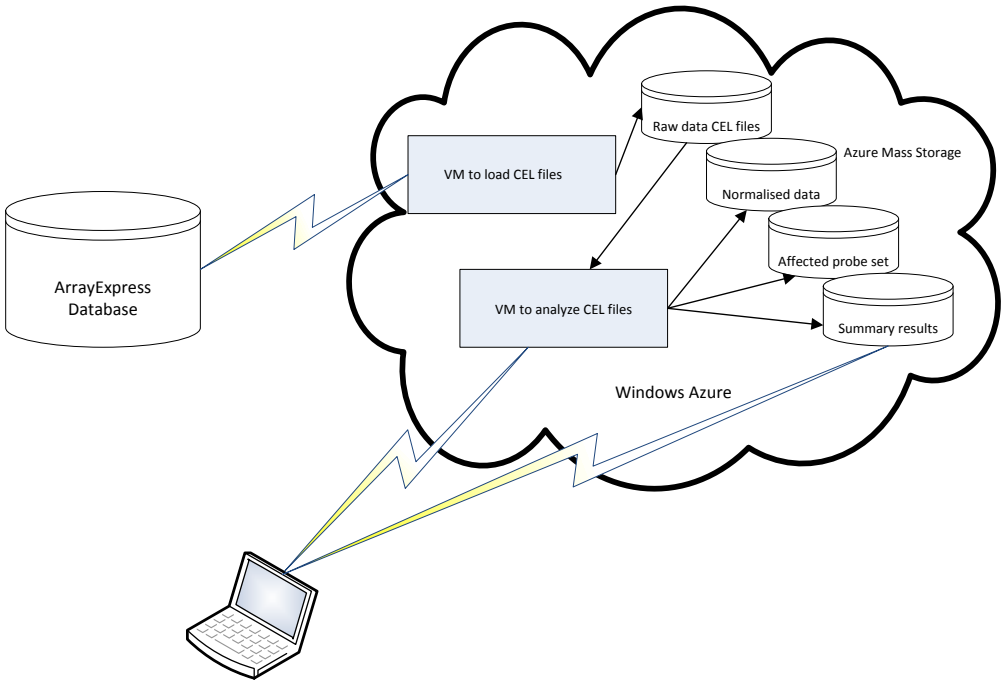


Figure 7.1: Flow of Data Analysis jobs

Figure 7.1 shows the basic flow of data and programs used in all the analyses performed on the Azure cloud. The ArrayExpress [36] Database was accessed over the internet using an R script to upload all possible HG_U133A CEL file datasets to Azure blob storage. These are labelled 'Raw data CEL files' in the Figure. The normalization and bias identification analysis scripts were run with parameters entered from a local PC, and results were stored in Azure blob storage prior to being downloaded for scrutiny.

The previous chapter has described in detail how jobs were submitted to the Azure cloud, both before and after the tool GWydiR was developed. The validation of the service using the R script to calculate the average correlation of expression values was also described.

Three types of analysis used by this work to examine the bias that G-stacks can bring to microarray data have been described in section 3.6. Examples were also given in that

chapter of the use of human HG_U133A GeneChip data with those analysis methods. Those results will not be repeated here but they will be summarized.

7.3 Results of analyzing human GeneChip data

In studies performed by Shanahan *et al.* on G-stacks in human HG_U133A microarray data [38], it was shown that about 15% of the 176 experiments tested are susceptible to significant G-stack bias. Now with microarray datasets potentially available for over 600 HG_U133A experiments, the Azure cloud facilities of storage space and processing time gave opportunity to perform a large scale analysis to check the effects of G-stacks more widely. In practice, the number of experiments with usable data was 573. The reasons for the number of experiments with usable data being reduced from over 600 to 573 are the many issues which prevented datasets being downloaded successfully, explanations for which are given in subsection 6.7.1.

7.3.1 Results using the average correlation of expression levels method

The first analysis described was the average correlation of expression levels over all CEL files of an experiment using a sample of 1000 G-stack probes. The results of calculating this average correlation for G-stack probes and then for C-stack probes, as a control, gave rise to Figure 3.10. This clearly showed that the average correlation for the C-stacks was closer to zero than for the G-stacks. The average correlation for the G-stacks ranged from 0.00 to 0.76, with 34 of the 176 experiments analyzed (i.e. 19%) showing an equal or larger average correlation for the G-stacks than 0.41. This value was described by Shanahan *et al.* [38] as being one where an experiment still exhibits a noticeable bias, whereas experiments with a smaller value for their average correlation of G-stack probes do not exhibit a noticeable bias. Figure 7.2 shows the GSE experiments with the highest actual values of the average G-stack probe correlations, sorted in order of largest values first. The figure shows the number of

GSE	No of CEL files	Average G-stack probe correlation
GSE1869	25	0.76
GSE6596	26	0.70
GSE1323	6	0.59
GSE3307	121	0.59
GSE5389	21	0.58
GSE3771	6	0.56
GSE1922	48	0.55
GSE4824	79	0.55
GSE5392	82	0.54
GSE3243	6	0.53
GSE5370	9	0.53
GSE5450	6	0.51
GSE5457	8	0.51
GSE4133	6	0.50
GSE1935	24	0.48
GSE6365	90	0.48
GSE2395	20	0.48
GSE1295	24	0.48
GSE5967	21	0.47
GSE6401	102	0.46
GSE3737	8	0.46
GSE5388	61	0.46
GSE3167	60	0.46
GSE1786	24	0.45
GSE6344	20	0.45
GSE2248	6	0.44
GSE873	5	0.44
GSE6914	20	0.43
GSE2044	13	0.43
GSE1518	8	0.43
GSE4636	18	0.43
GSE1729	43	0.42
GSE5258	346	0.41
GSE2603	121	0.41
GSE994	75	0.40
GSE3794	6	0.40
GSE1133	158	0.40
GSE2742	27	0.39
GSE3419	16	0.38
GSE3846	108	0.38
GSE6011	37	0.38
GSE2487	10	0.38
GSE4045	37	0.38
GSE3780	30	0.37
GSE1650	30	0.37
GSE3585	12	0.37
GSE1317	3	0.36
GSE4885	12	0.36
GSE3524	20	0.35
GSE6184	33	0.34
GSE6015	15	0.34
GSE2018	34	0.34
GSE6613	105	0.34
GSE4646	23	0.34
GSE1297	31	0.34
GSE5418	71	0.33

Figure 7.2: The average correlation of expression levels for G-stack probes over all CEL files of the designated experiments, and the number of CEL files in each case, ordered by highest correlation first

CEL files used in each experiment which varies quite widely from 3 to 346. This raised the question as to whether the number of CEL files in an experiment has any significant impact on the expression levels detected. This question will be addressed in section 7.4.

7.3.2 Results using the summarised expression data

The second analysis described was the summarized expression data. Figure 3.12 shows the expression data for 573 experiments using HG_U133A GeneChip data showing median differences when C-stacks and G-stacks are removed respectively. This gives a clear visualization of the difference between the effects of removing C-stack probes which remain heavily concentrated around the origin, and the effect of removing G-stack probes which indicate that those probes were exhibiting a bias.

7.3.3 Results using the correlation with each other of probe sets containing G-stacks

The third analysis described was to test the extent to which probe sets containing G-stacks are correlated with each other. Normally it is expected that data from probes within the same probe set would be correlated with each other because they should be regulated by the activity of the same gene. Figure 3.13 shows the results of this analysis where the evidence is clear that probe sets containing G-stack probes are correlated with each other. The graph also shows that probe sets containing C-stack probes are not correlated with each other.

The second and third analyses just summarized were both carried out using RMA normalization. In section 7.5 the effect of using PLIER instead of RMA for the normalization step will be investigated.

7.4 Significance of the number of CEL files in an experiment

The number of CEL files of raw data from an individual experiment can vary from one to several thousand. A question was asked about whether the number of CEL files in an individual experiment affected the results of the analyses in this work. It was decided to test whether results varied depending on the number of CEL files in experiments. To this end, the results from the correlation analysis were grouped into ranges according to the number of CEL files, with the resultant histograms plotted in Figure 7.3.

The six charts in Figure 7.3 each show a different range of numbers of CEL files: 6 to 10, 11 to 20, 21 to 30, 31 to 50, 51 to 100 and 101 to 200. The ranges were chosen to give a number of GSEs in each range, and the numbers are shown in the chart titles. It can be seen that for most of the CEL file ranges the largest frequency of their GSE GT correlation median differences lie between -0.10 and -0.05. The 51 to 100 CEL file range shows its largest frequency to be less than -0.10, as does the 11 to 20 range. However, visual inspection of the six histograms which comprise Figure 7.3 does not show any strong results for GT correlations, so that the number of CEL files in an experiment is not believed to be significant.

To confirm this hypothesis more rigorously the basic descriptive statistics of the GT correlation data are presented for each of the six ranges of numbers of CEL files used in Figure 7.3. Table 7.1 shows the mean, median, standard deviation and the maximum and minimum figures for the median differences of GT correlation data that are plotted in Figure 7.3.

It can be seen from Table 7.1 that the mean and the median for each range of CEL files is between -0.09 and -0.12 which is a narrow range of values. The standard deviation of all groups is 0.04 or 0.05 which is again similar to each other. The range of CEL files from 21 to 30 had a lower maximum at -0.05 than other groups, and the range of CEL files from

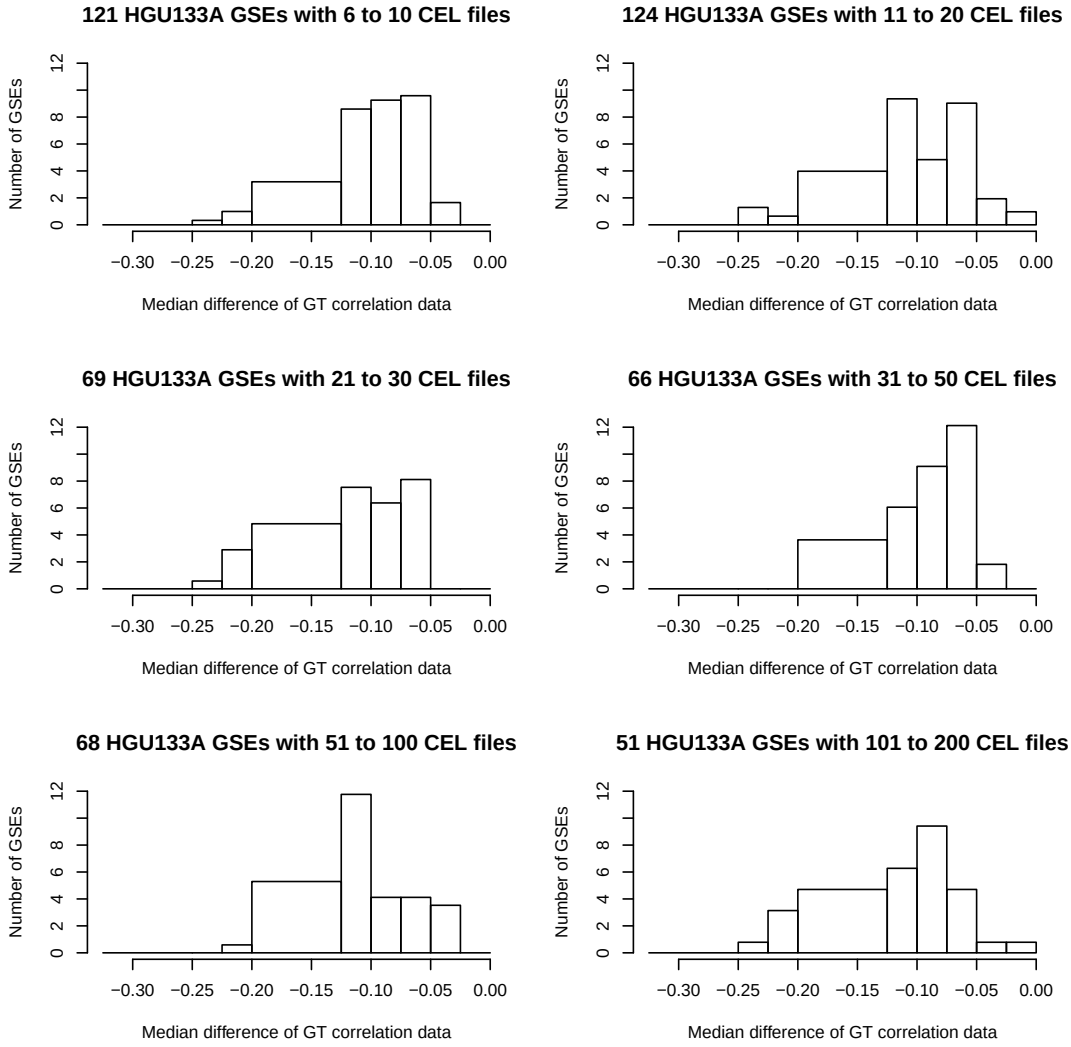


Figure 7.3: Plots of the median differences in GT correlation data (more than 0.4) over 573 HG_U133A GSEs, in ranges depending on the number of CEL files in each GSE

No. CEL files	Mean	Median	Std Dev	Maximum	Minimum
6-10	-0.10	-0.09	0.04	-0.03	-0.24
11-20	-0.11	-0.10	0.05	-0.02	-0.24
21-30	-0.12	-0.12	0.05	-0.05	-0.23
31-50	-0.09	-0.09	0.04	-0.03	-0.17
51-100	-0.11	-0.12	0.04	-0.03	-0.21
101-200	-0.12	-0.11	0.05	-0.02	-0.23

Table 7.1: The basic descriptive statistics of the GT correlation data for each range of CEL files

31 to 50 had a higher minimum at -0.17 than other groups. With these basic descriptive statistics showing that each range of CEL file numbers has results of the same order, the hypothesis is confirmed that the number of CEL files in an experiment is not significant.

7.5 Significance of using the PLIER method of normalization

The analyses described so far have used the RMA method of normalization. In this section the PLIER method is compared and contrasted with the use of RMA. The PLIER method takes longer to compute than the RMA method. In comparing GT correlation data for RMA and for PLIER it is seen that PLIER appears to go a long way to correcting the bias introduced by G-stacks, see subsection 7.5.2.

7.5.1 Extra time taken for PLIER vs RMA

The third type of analysis (section 7.3.3) was used in comparing the length of time that PLIER jobs took compared with RMA jobs. Figure 7.4 shows the processing time for particular experiments plotted against the number of CEL files in each experiment. The dashed line pertains to experiments processed using the RMA method of normalization, and the solid line shows the same experiments processed using the PLIER normalization method. The time taken for the RMA experiments varied from about one minute when processing two or four CEL files to 17 minutes when processing 128 CEL files. The time taken for the PLIER experiments varied from about one minute when processing two or four CEL files to an average of two hours and 25 minutes when processing 128 CEL files.

In particular the groups of experiments having 2, 4, 8, 16, 32, 64 and 128 CEL files were chosen for a graph to see the relationship between the number of CEL files and the processing time. The solid and dashed lines are shown between the means of each group of points. It can be clearly seen that the relationship is approximately linear for the RMA

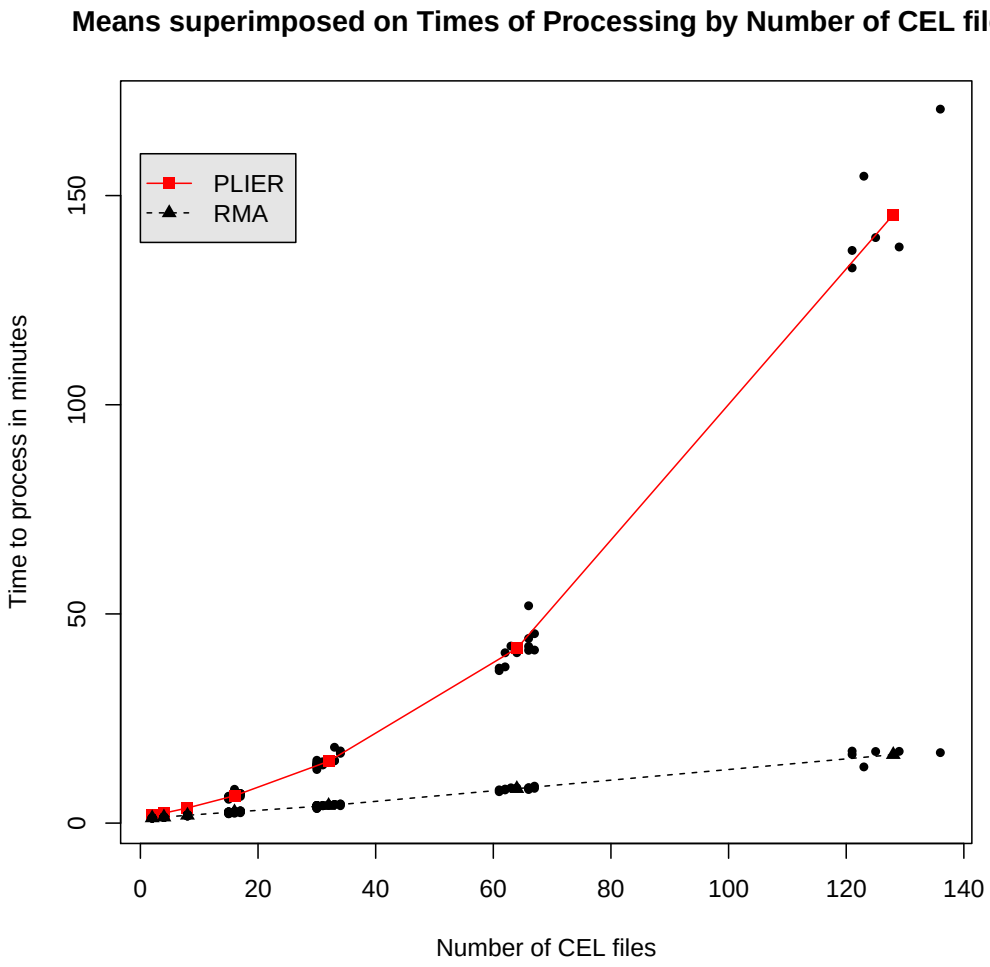


Figure 7.4: Plots showing the length of processing time for RMA compared to PLIER normalization routines, on experiments with 2, 4, 8, 16, 32, 64, and 128 (approx.) CEL files. The mean of the timing of each group of experiments is shown by a triangle (RMA) and by a square (PLIER).

normalization method, but for PLIER it is seen that as the number of CEL files doubles, the processing time increases more than twofold. The reason is due to the PLIER algorithm having to process the CEL file data more than once.

Where cost is an issue for processing large amounts of data on a cloud, one must take into account the normalization method used.

7.5.2 Analysis of G-stack data using PLIER compared to using RMA

It was found that the use of the PLIER normalization method tended to concentrate the results of analyses within smaller bounds, whereas the RMA method gave results that were more spread. This is discussed in depth in section 7.6.

It has been shown by Qu *et al.* [67] that compared to RMA summarization, PLIER summarization can lead to over-estimation of gene-level expression changes, relative to exon-level expression changes in two-group comparisons. This work was done by using the published human tissue panel dataset (containing 11 different human tissues) based on the Affymetrix Human Exon 1.0 ST array. The conclusion of Qu *et al.* is that PLIER and RMA behave differently in the detection of alternative splicing events. The observed tendency of PLIER to "detect relatively skipped exons on up-regulated genes and relatively included exons on down-regulated genes" was due to technical bias in their opinion. Gaidatzis *et al.* [68] also argue that systematic bias persists due to sequence features of the probes in Affymetrix exon arrays, meaning that detection of alternative splicing events needs correction for which they offer software.

The analysis in this work was not done on an Affymetrix human exon array, but on the human HG_U133A array. Considering the marked difference in the spread of the bars in Figure 7.5 using PLIER as compared with using RMA, the question arose: how exactly do they compare?

Figure 7.6 is drawn from the same data as Figure 7.5 but by plotting the median differences in correlation values using RMA against those using PLIER. It was expected that

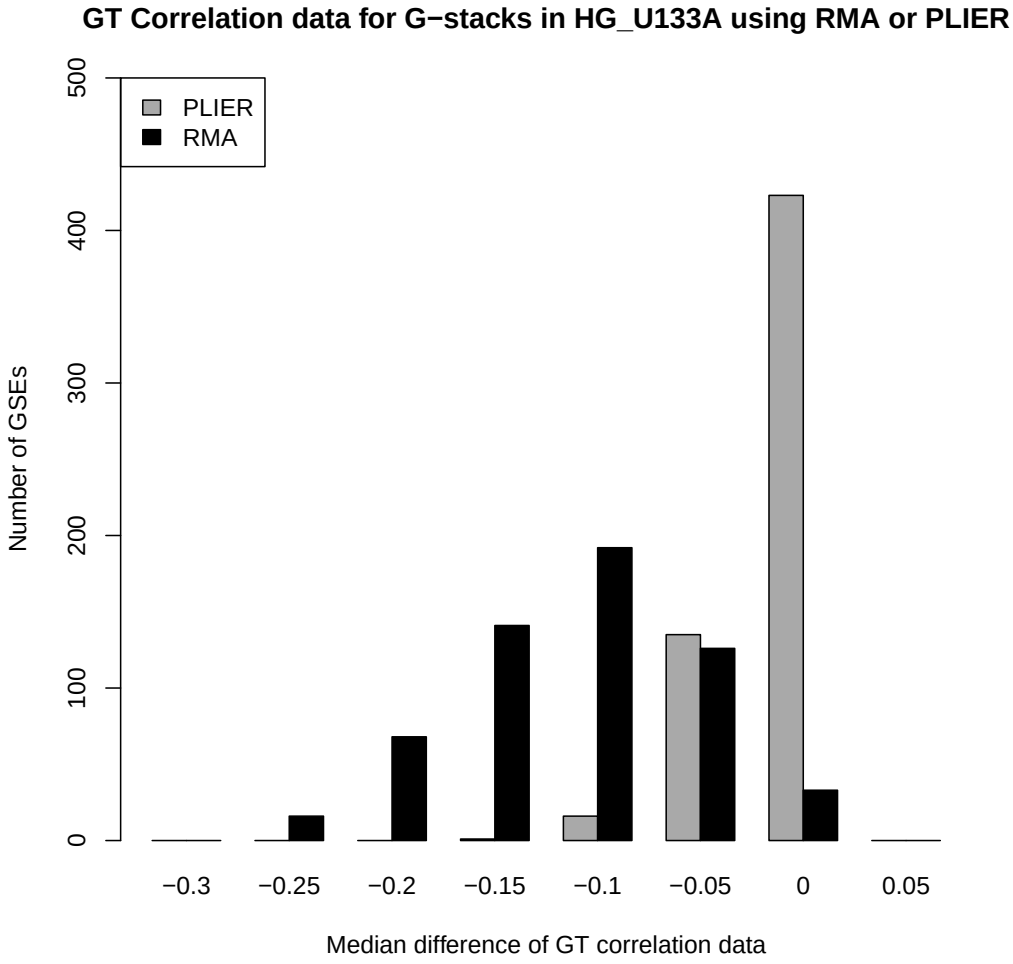


Figure 7.5: Median differences in correlation values (greater than 0.4) between where G-stacks were kept in and removed, using RMA and PLIER for all 576 experiments

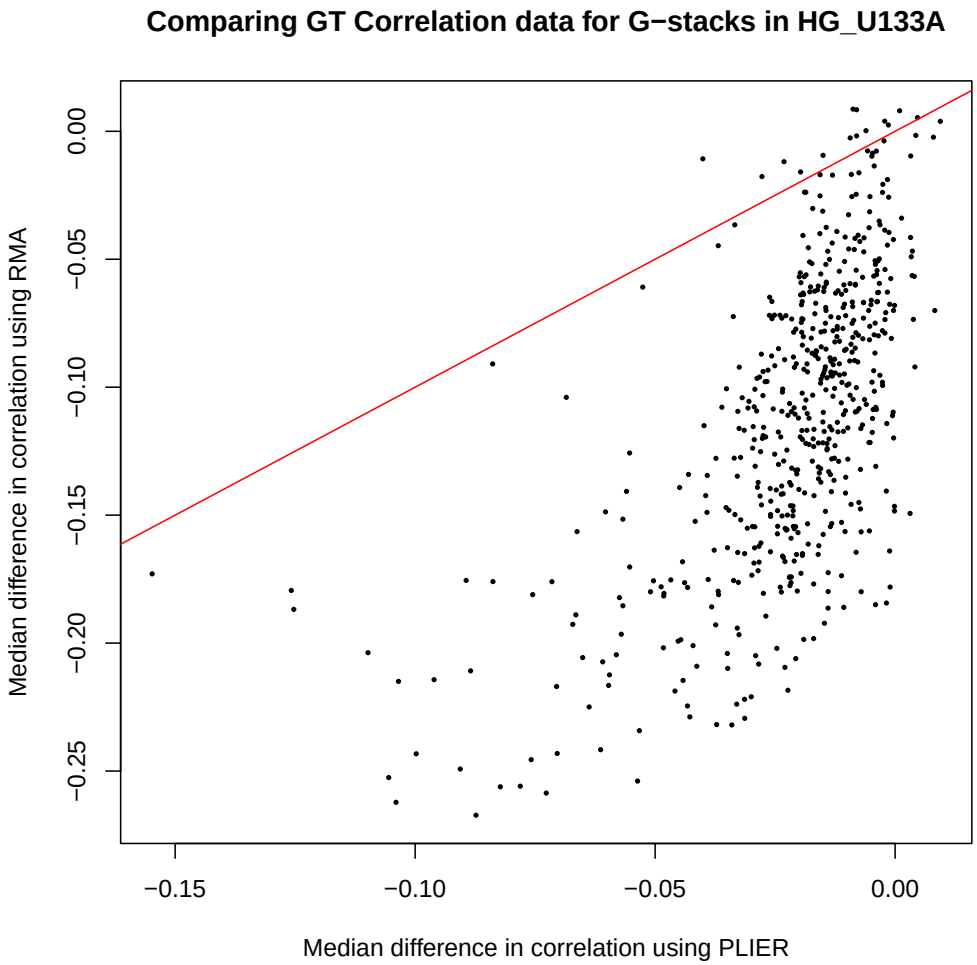


Figure 7.6: Comparison of median differences in correlation values (greater than 0.4) between where G-stacks were kept in and removed, using RMA and PLIER for all 576 experiments. The line of equality is shown.

they would be correlated as they are two different methods of achieving the normalization of the same expression data, yet the correlation between them is only 0.664. The RMA median shift is generally larger than that for PLIER. This is consistent with the contention that PLIER achieves improvement over RMA for ameliorating the bias caused by G-stacks.

7.6 Further analysis of HG_U133A expression data

Microarrays such as HG_U133A are concerned with probes which are organised into probe sets to reveal the expression of particular genes. Here we will compare the expression data of G-stacks and C-stacks in the HG_U133A microarray when RMA and when PLIER normalization methods are used. Then we will consider probe sets to see whether probe sets are more correlated with each other in the case of G-stacks or in the case of C-stacks.

7.6.1 Expression data of HG_U133A for G-stacks and C-stacks using RMA and PLIER

The darker columns of the top graph in Figure 7.7 show a chart of the medians of the difference in expression data between having two or more G-stack probes left in and having G-stack probes removed, where the RMA normalization method (see subsection 3.3.3 on page 49) was used. If there were no bias in the data due to the G-stacks, then this chart would show the median difference values equally distributed around zero. However, it can be seen that more of the median values for the G-stacks are less than zero than are greater than zero, which confirms the anticipated bias effect. The C-stack columns in a lighter shade act as a control as C-stacks do not show a similar bias, according to Shanahan *et al.* [38].

The lower graph of Figure 7.7 shows the results from the same analysis of both G-stacks and C-stacks, but using the PLIER normalization method (see subsection 3.3.4 on page 50). Shanahan *et al.* [38] found that PLIER was able to "ameliorate the bias due to G-stacks" and that the commonly employed normalizations (MAS5, RMA and gcRMA) were susceptible

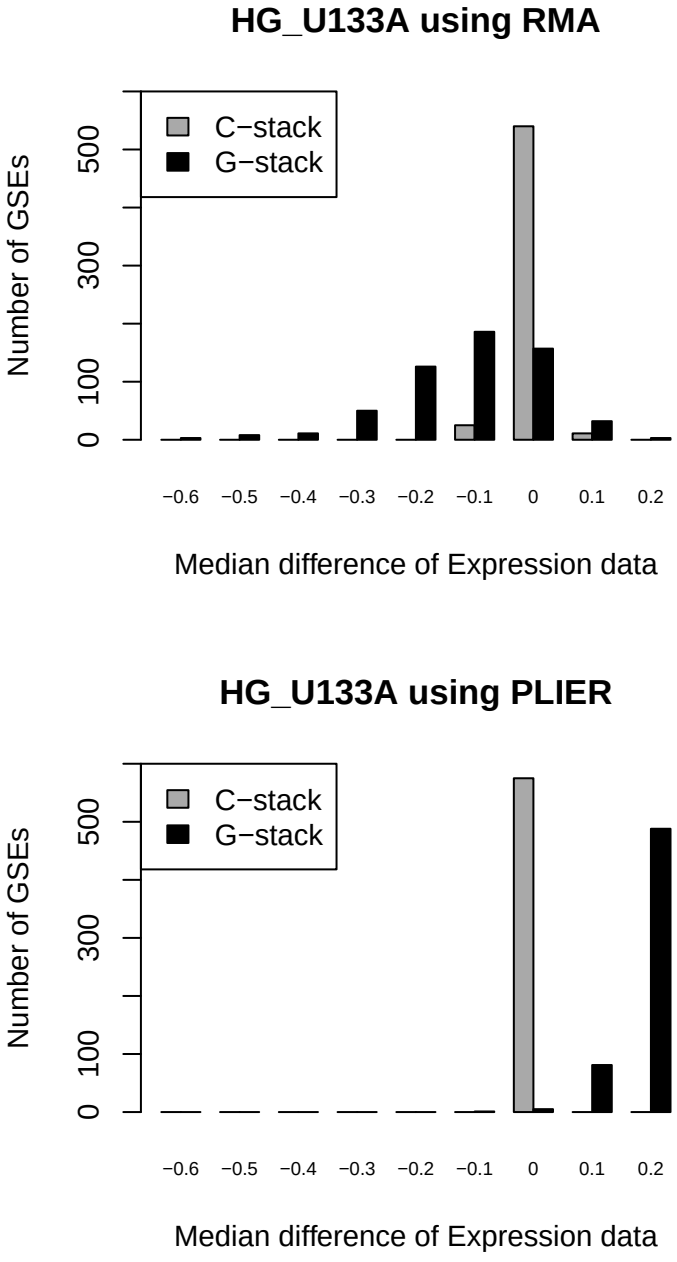


Figure 7.7: Expression estimates for HG_U133A GeneChip

to significant G-stack bias. Memon *et al.* [69] showed that G-stack probes on other types of GeneChip for various mammals exhibited high correlations with each other, and thus it is likely that they would exhibit a bias in normalized data too. This work confirms this view, and demonstrates how PLIER can improve the bias from G-stacks in gene expression data.

A similar effect of using PLIER normalization will be seen when analyzing the expression data from the human HG_U133_Plus2 GeneChip in section 8.3.1. Although ameliorating the bias due to G-stacks in probes, the PLIER method can perhaps overcompensate for G-stacks as shown by the black bars on the lower chart of Figure 7.7 being moved to the positive side of the origin.

7.6.2 HG_U133A probe set to probe set correlation data

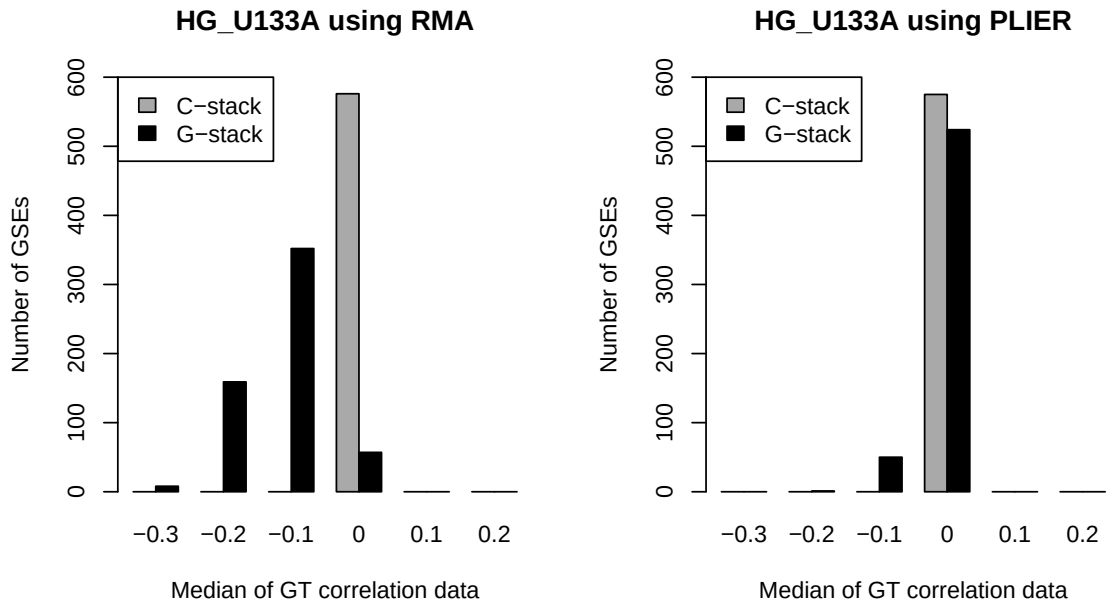


Figure 7.8: Correlation differences for HG_U133A GeneChip

The left-hand graph in Figure 7.8 shows a bar chart of the median differences in probe set to probe set correlation data between having two or more G-stack probes left in and having G-stack probes removed, where the RMA normalization method was used. The shorthand GT is used to denote that correlations greater than 0.4 were used here (see a fuller

explanation in section 3.6.3). The G-stack probe sets were more highly correlated with each other than were the C-stack probe sets. This can be seen by the spread of the darker bars of the chart being seen further from the origin than the lighter bars. The fact that they are negative is only because of the way the difference was calculated. The difference between leaving the G-stacks in and taking them out could have had its subtraction done the other way round (i.e. the difference between taking out the G-stacks and leaving them in).

The right-hand graph in Figure 7.8 shows the equivalent analysis calculations using the PLIER normalization routine instead of RMA. There are still a small number of experiments (about 10%) affected by a small amount of G-stack bias, as shown by the small spread of some G-stack median differences from the origin. However, for the majority of experiments, the use of PLIER appears to correct the G-stack bias demonstrated with the use of RMA.

Chapter 8

Wide scale Survey

8.1 Introduction

The previous chapter has described how the Windows Azure cloud was used to perform an investigation of all the available Affymetrix GeneChip HG_U133A datasets to establish how widespread is the bias in the datasets due to G-stacks. The next stage in the Venus-C project was to extend this wide scale survey to another GeneChip HG_U133_Plus2 of *H. sapiens* and to GeneChips of other species: *Arabidopsis thaliana* and *Pseudomonas aeruginosa*.

It has already been explained that the results of the G-stack survey of 176 experiments using the human GeneChip HG_U133A bore out the results found by Shanahan *et al* [38]. The first experiments using Windows Azure were to check that the R scripts obtained similar results again, which they did. These results were shown at a Poster Presentation at the ISMB/ECCB Annual International Conference in July 2013 in Berlin (see Appendix F).

At the end of the previous chapter the analysis of G-stack data was described using the PLIER normalization pipeline and contrasted with the use of RMA normalization. The example of GT correlation data was used. The use of both RMA and PLIER is shown in this chapter on a wide scale survey of microarray data across the range of GeneChips mentioned above. The data for expression levels and for GT correlations will be explored. First the scatter plots of the expression data of G-stacks compared with that of C-stacks will

be considered.

8.2 Analysis of expression levels across species using scatter plots

The data on expression levels for the GeneChip HG_U133A has already been discussed and the effect of G-stacks has been shown and contrasted with the effect of C-stacks. Here in Figure 8.1 the median shift in G-stack expression data has been plotted against the median shift in C-stack expression data, using a scatter plot. The left hand plot was obtained using RMA and the right hand plot used PLIER. The spread of points in the PLIER plot is confined much closer to the origin of the graph than in the RMA plot. This effect is discussed in greater detail in section 7.6 and it appears in these scatter plots that PLIER largely overcomes the bias effect of G-stacks seen in the RMA graphs.

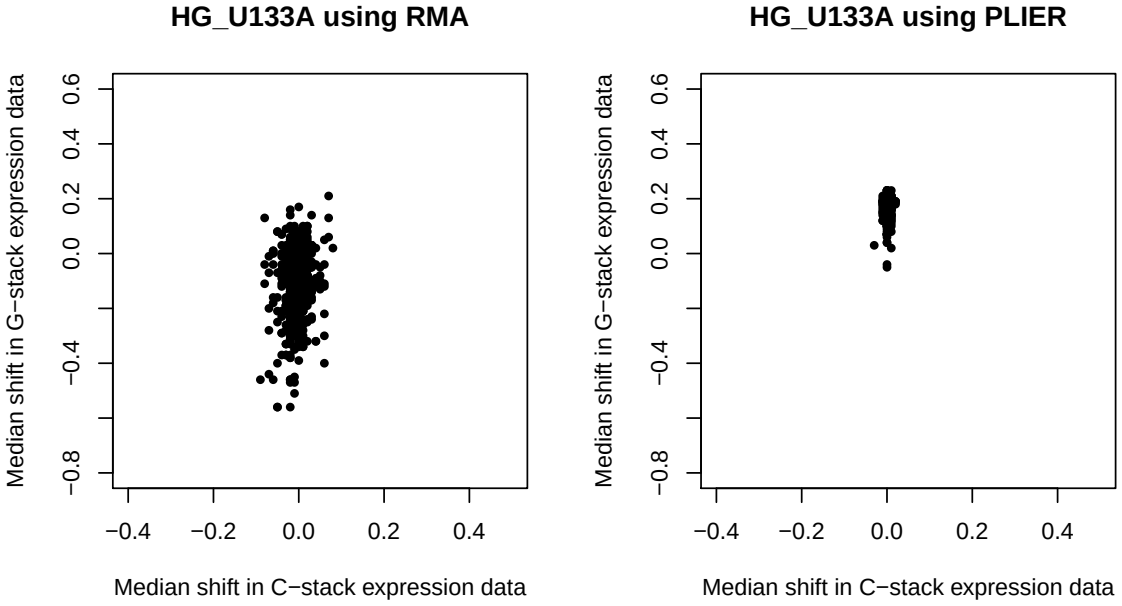


Figure 8.1: Scatter Plot of Expression Data for 576 datasets of HG_U133A GeneChip

In the RMA plot it can be seen that the points are spread more vertically than horizontally because the G-stack expression data varies more from the origin than the C-stack data. The

G-stack expression data is skewed towards the negative, because of the bias shown in the median difference between retaining and removing probes with two or more G-stacks.

In the case of the second human GeneChip that was analyzed: HG_U133_Plus2, the scatter plot of expression data in Figure 8.2 shows a wider spread of points, particularly in the RMA plot. Out of more than 2000 experiments that were available for this GeneChip, data from 1999 experiments were successfully uploaded to the cloud and analyzed.

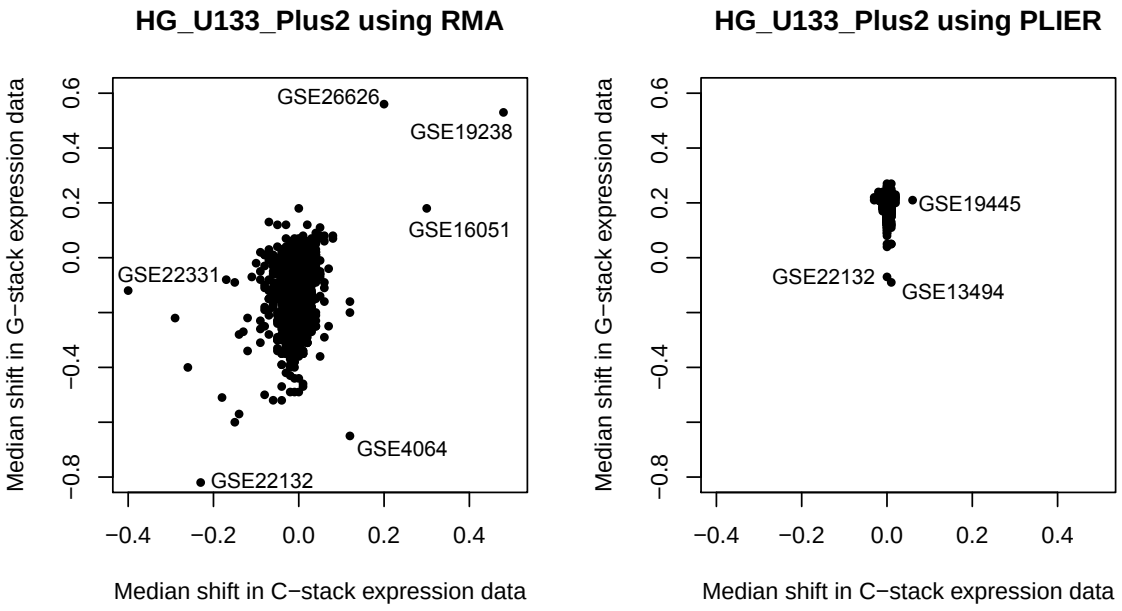


Figure 8.2: Scatter Plots of Expression Data for 1999 datasets of the HG_U133_Plus2 GeneChip

With over three times as many datasets available for this GeneChip as for HG_U133A, it may not be surprising that a few more outliers show up in this case. It is apparent that each of the species that have been investigated in this study show a few outliers.

The RMA scatter plot for HG_U133_Plus2 in Figure 8.2 shows a few pronounced outliers which is not a feature of the HG_U133A plot at this scale. Table 8.1 shows the information known about these outliers. The first three experiments in the table are the three in the top right hand corner of the RMA scatter plot, and they all have high values for both G-stacks and C-stacks. The two outliers which are lowest in the RMA plot of Figure 8.2 have unusually low values for the median shift of G-stacks, and GSE22331 has an unusually

Experiment	Median shift for G-stacks	Median shift for C-stacks	No. of CEL files	Experiment Type	Experiment Subject
GSE26626 R	0.56	0.20	4	Expression profiling by array	mRNAs associated with human Pumilio2 protein (PUM2)
GSE19238 R	0.53	0.48	2	Expression profiling by array	Expression data for 2 obese subjects from the SibPair cohort with a deletion on 16p11.2
GSE16051R	0.18	0.30	4	Expression profiling by array	Effect of ectotopic expression of K13 on global gene expression in HUVEC
GSE4064 R	-0.65	0.12	4	Expression profiling by array	Gene expression of growing antler in Cervus elaphus (deer)
GSE22132 R	-0.82	-0.23	2	Expression profiling by array	Expression data from purified human platelets
GSE22331 R	-0.40	-0.12	2	Expression profiling by array	Expression data from ejaculated spermatozoa of normozoospermic and asthenozoospermic men
GSE19445 P	0.21	0.06	9	Transcription profiling by array	Androgen-induced TOP2B-mediated double-strand breaks and prostate cancer gene rearrangements
GSE22132 P	-0.07	0.00	2	Expression profiling by array	Expression data from purified human platelets
GSE13494 P	-0.09	0.01	4	Transcription profiling by array	Expression data from human saliva exosomes

Table 8.1: Details of the outliers in the scatter plots of HG_U133_Plus2. Outliers are listed from the top as labelled clockwise in the plots of Figure 8.2. The capital letters 'R' and 'P' after the GSE experiment number stand for RMA and PLIER respectively. GSE22132 is an outlier in both the RMA and PLIER plots.

low value for the median shift of C-stacks, but is normal for its G-stacks. Table 8.1 shows a range of tissue types used in the experiments and shows no common features across the experiments except that the number of CEL files in the experiment is either two or four for the RMA outliers and for all but one of the PLIER outliers. It is unlikely that this number of CEL files has any link with the fact that these six experiments are outliers, because there are over a hundred other experiments with two or four CEL files which are not outliers. However, it may be that those particular tissue samples combined with the small number of CEL files were significant in creating outliers. The experiment GSE22132 shows as an outlier in both the RMA and the PLIER plots. It has the lowest value for the median shift in G-stack expression data in the RMA case, and the second to lowest value in the PLIER case. It seems that PLIER normalization could not ameliorate the value into the main range of values for this particular experiment.

The difference between using PLIER and RMA to calculate the gene expression values is again shown clearly in the analysis of HG_U133_Plus2 expression data in Figure 8.2. The use of PLIER shows a very marked concentration of the values into a small area. This bears out the contention of Therneau and Ballman [50] that PLIER improves the gene expression estimate that was mentioned in section 3.4.1.

The next two figures (Figure 8.3 and Figure 8.4) are comparisons of RMA and PLIER in the scatter plots of expression data for a plant species (Arabidopsis) and a bacteria species (Pseudomonas). These plots can be compared with those for the two human GeneChips already discussed. The Arabidopsis data was analyzed from 625 microarray experiments and the plots are shown in Figure 8.3. They show in the right hand graph that the PLIER normalization causes the median shift in G-stacks to be greater, and in a positive direction compared to the median shifts in C-stacks. The RMA normalization shown in the left hand graph has a broadly similar grouping to the human chips, with some outliers.

The outliers have been investigated and the results are shown in Table 8.2. Again there is apparently no similarity to be spotted in the type of tissue being analyzed. There is no

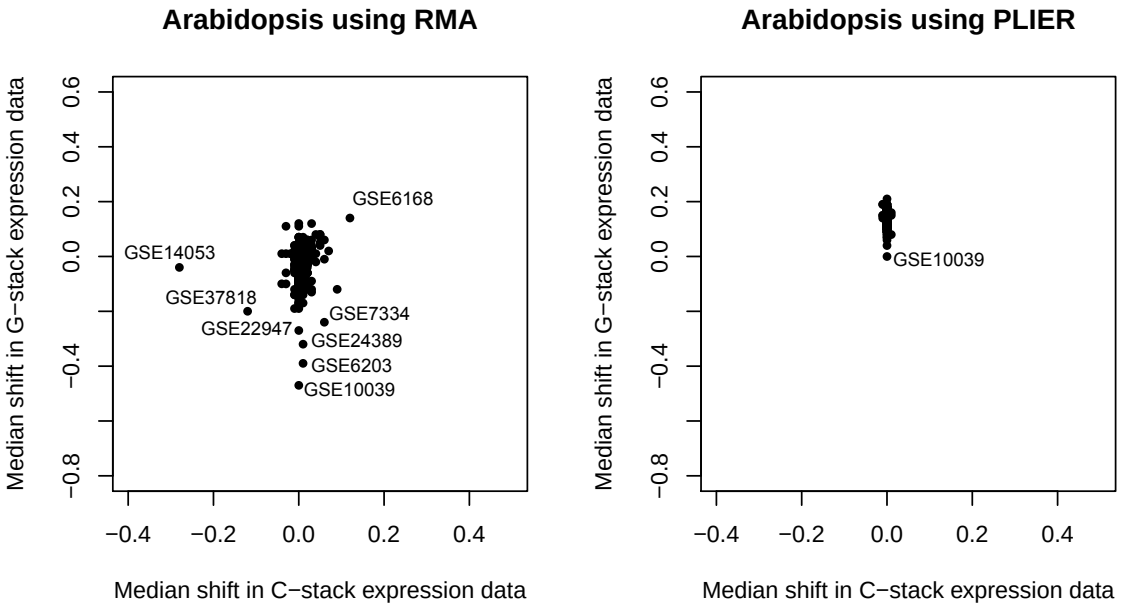


Figure 8.3: Scatter Plot of Expression Data for 625 datasets of Arabidopsis GeneChip

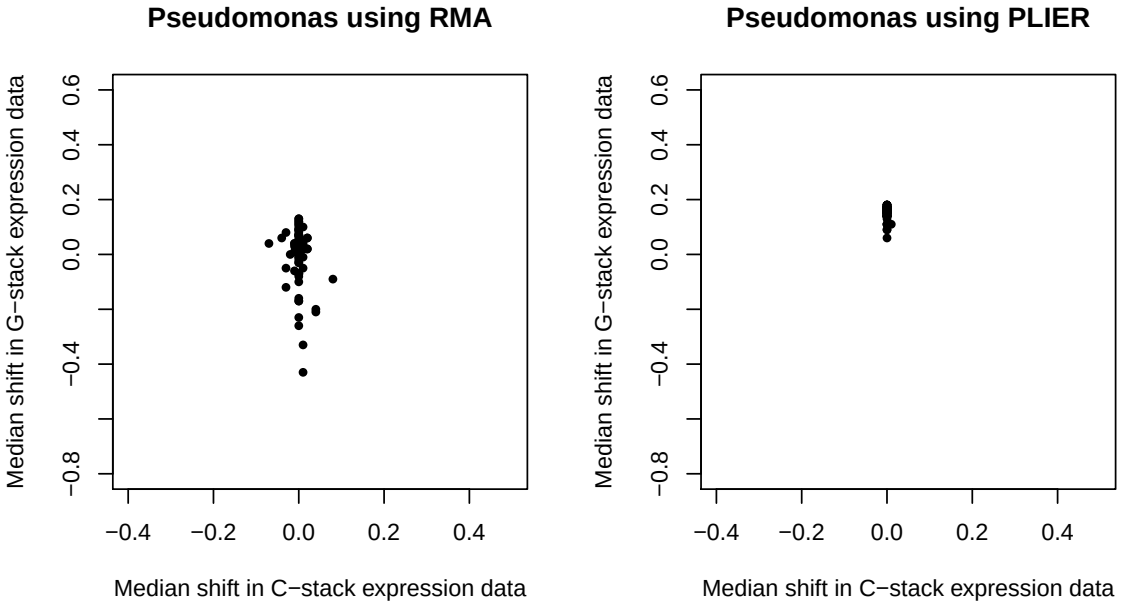


Figure 8.4: Scatter Plot of Expression Data for 79 datasets of Pseudomonas GeneChip

Experiment	Median shift for G-stacks	Median shift for C-stacks	No. of CEL files	Experiment Type	Experiment Subject
GSE6168 R	0.14	0.12	2	Transcription profiling by array	Comparison of the transcript profile of the auxin resistant axr4 mutant and wild-type Col0
GSE7334 R	-0.24	0.06	4	Transcription profiling by array	Transcription profiling of Arabidopsis response to aluminum stress
GSE24389 R	-0.32	0.01	7	Analysis of Aloe vera transcriptome	Aloe vera transcriptome was analysed by hybridising samples of root and leaf tissue to Arabidopsis ATH1 array
GSE6203 R	-0.39	0.01	13	Transcription profiling by array	Transcription profiling of shoot tissue from 12 different accessions
GSE10039 R	-0.47	0.00	8	Transcription profiling by array	Shoots from plants grown with low Molybdenum (Mo)
GSE22947 R	-0.27	0.00	24	Expression data from in-vitro induced Interfascicular Cambium	In-vitro induced establishment and activity of the interfascicular cambium in stems under auxin treatments
GSE37818 R	-0.2	-0.12	4	Expression data in WT and erf6 mutant	Expression data in WT and erf6 mutant under high-light treatment
GSE14053 R	-0.04	-0.28	2	Expression data from trichomes and pavement cells	Comparison of cell sap from trichomes and pavement cells
GSE10039 P	0.00	0.00	8	Transcription profiling by array	Shoots from plants grown with low Molybdenum (Mo)

Table 8.2: Details of the outliers in the scatter plots of Arabidopsis. Outliers are listed from the top as labelled clockwise in the plots of Figure 8.3. The capital letters 'R' and 'P' after the GSE experiment number stand for RMA and PLIER respectively. GSE10039 is an outlier in both the RMA and PLIER plots.

pattern in the number of CEL files in the outlier experiments, and only four of the nine experiments had two or four CEL files in this case. There is one experiment, GSE10039, which is an outlier in both the RMA and the PLIER plots. Again it is the experiment which has the lowest value for the median shift in G-stack expression data in the RMA plot and the lowest value in the PLIER plot.

The Pseudomonas data was analyzed from 79 microarray experiments and the plots are shown in Figure 8.4. Again the scatter plots show similar features to those for the other species. The points in the PLIER plot are more focussed and close to each other, and those in the RMA plot are more spread out with some outliers. The outliers were investigated and again it was noted that one experiment, GSE27674, showed very low values for the median shift in G-stack expression data on both the RMA and the PLIER plots.

8.3 Comparison of HG_U133A and HG_U133_Plus2 microarray data

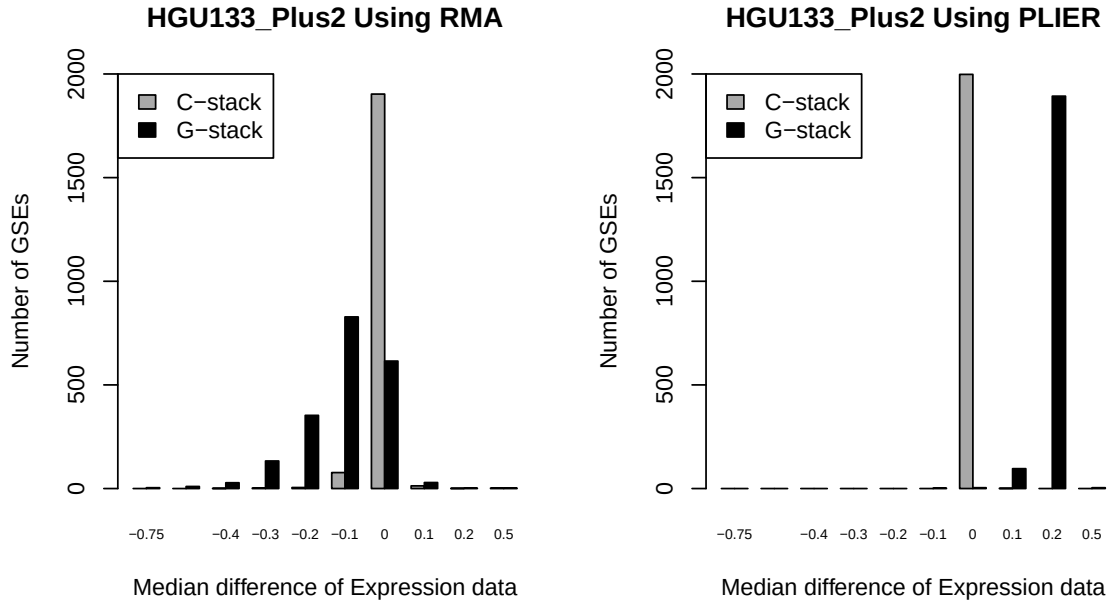


Figure 8.5: Expression Data for HG_U133_Plus2 GeneChip

8.3.1 HG_U133_Plus2 expression data

The first consideration with the data from the second human GeneChip HG_U133_Plus2 was to compare the expression data results from using RMA and the PLIER normalization methods. In Figure 8.5 it is seen that the effect of PLIER has been to overcompensate for G-stacks so that the median difference between keeping and removing probe sets with two or more G-stacks has moved mainly to the 0.2 bin which is large compared with using RMA. This effect was also seen in Figure 7.7 for the HG_U133A GeneChip.

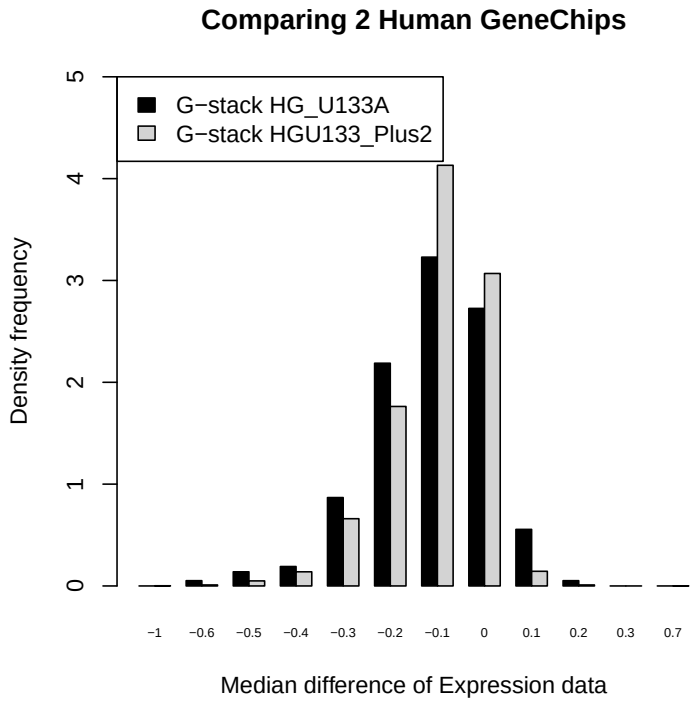


Figure 8.6: Comparing G-stacks in HG_U133A and HG_U133_Plus2 GeneChips

The effect of G-stacks are compared for two human GeneChips in Figure 8.6, using the median difference of expression data. The chart bins used are the same as in previous graphs, but instead of numbers of GSEs the y-axis shows the density frequency of the two distributions. This effectively allows the comparison of the two GeneChips despite them having different numbers of experiment data available.

The HG_U133_Plus2 GeneChip was designed later than the HG_U133A chip and more

account was taken of the sequences like G-stacks which could bias the biological data. These were avoided where possible in the chip design. The results in Figure 8.6 show that the HG_U133_Plus2 GeneChip expression data are less biased than the HG_U133A GeneChip data because the lighter bars on the Figure are grouped nearer to the origin than the dark bars. If the vertical axis figures were multiplied by ten, they would show percentages of the experiments represented in each bar.

8.3.2 HG_U133_Plus2 Probe Set to Probe Set Correlation Data

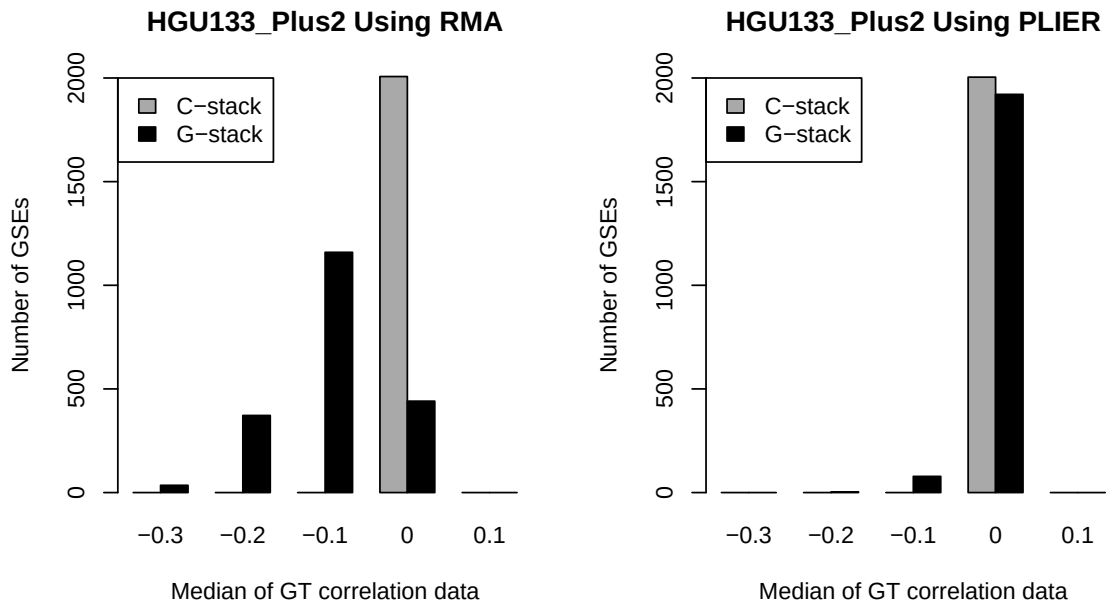


Figure 8.7: Comparison of GT correlation data for HG_U133_Plus2 GeneChip using RMA and using PLIER

Figure 8.7 shows two graphs side by side, one using RMA and the other using PLIER for the normalization step. The graphs show the frequency of the median difference of GT correlation data where every probe set is compared to every other probe set for the cases where probes containing two or more G-stacks have been removed or retained. The same situation for C-stacks is shown in the graphs as a control. It can be seen that the use of PLIER largely corrects the inherent bias in the data due to the G-stacks, whereas the

C-stacks do not show bias, and are represented similarly whether using RMA or PLIER.

8.4 Comparison of Arabidopsis and Human microarray data

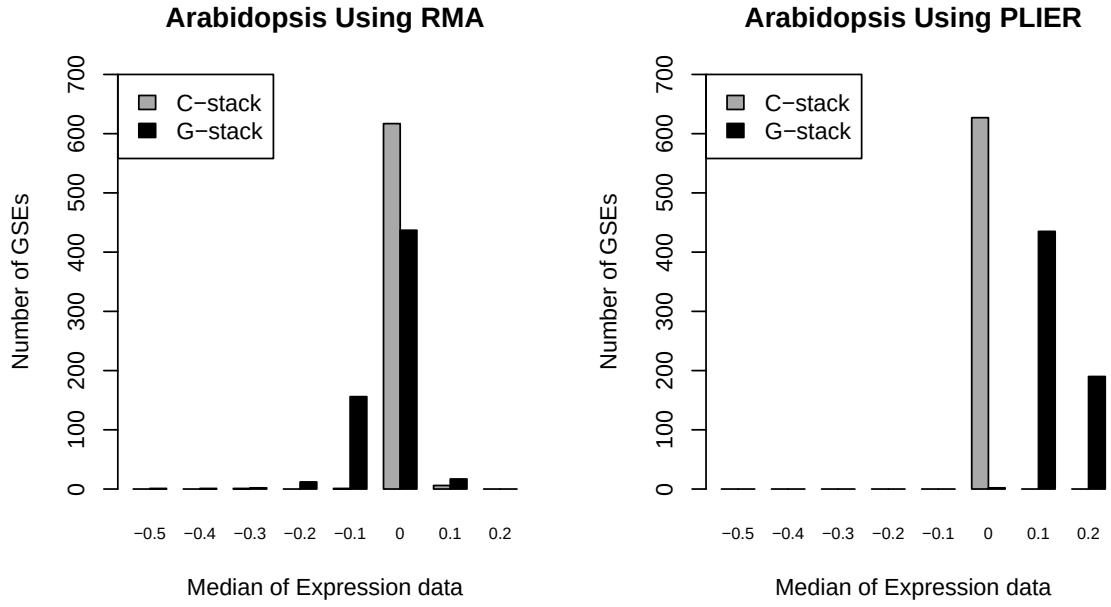


Figure 8.8: Expression data for Arabidopsis GeneChip

8.4.1 Arabidopsis expression data

The first consideration with the Arabidopsis microarrays was to compare the expression data using both RMA and PLIER. Figure 8.8 shows on the left using RMA that the G-stacks are spread more than the C-stacks which provide the control. This means that G-stacks in Arabidopsis are likely to be causing a slight bias in expression data. In the PLIER case on the right, where the black bars are all on the positive side of the origin, it appears that PLIER is overcompensating for possible G-stacks and introducing a bias itself.

It has been reported by Memon [70] that the Arabidopsis microarray data is less affected

by the bias due to G-stacks than both of the human microarray data chips examined here. This is confirmed by Figure 8.9 which plots the density frequency instead of the number of GSEs because the three chips had widely varying numbers of experiments to look at. The figure shows that the median difference of expression data (for two or more G-stack probes removed versus being left in) for Arabidopsis is more closely grouped around the origin which confirms Memon's report.

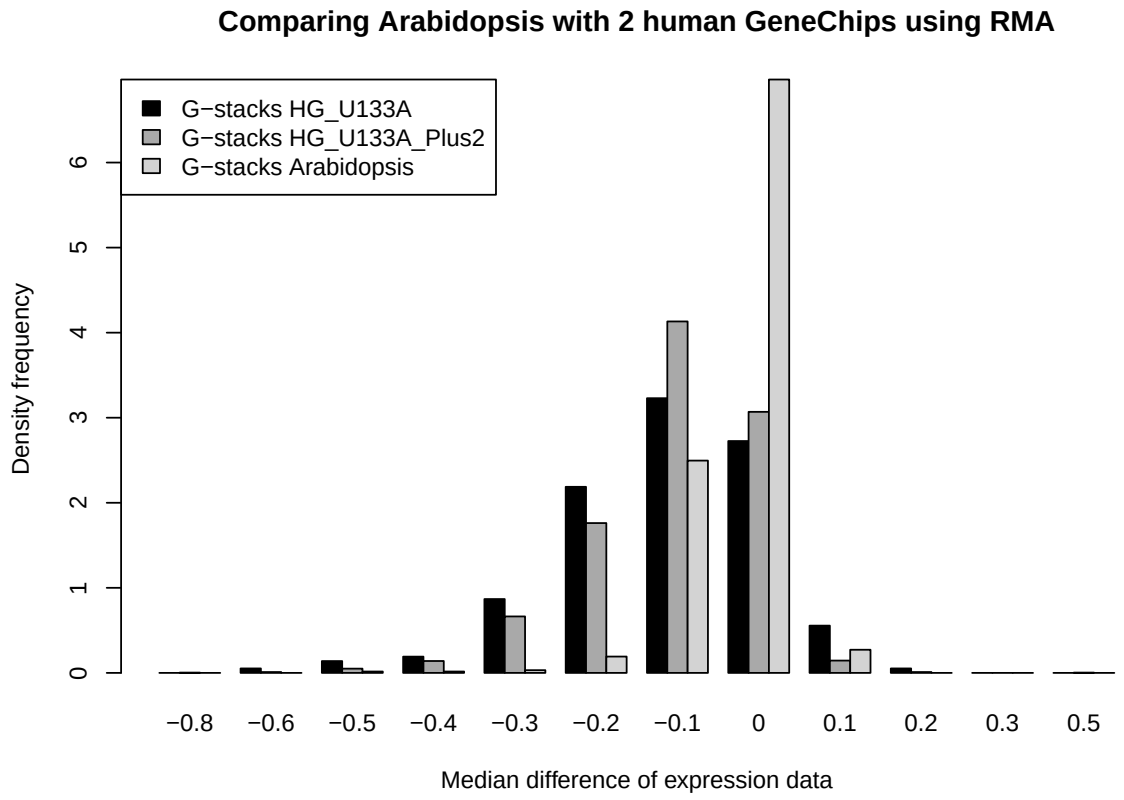


Figure 8.9: Comparing G-stacks in Arabidopsis GeneChip with those in HGU_133A and HGU_133_Plus2 GeneChips for expression data

No. of G-stack probes in a probe set	**0**	**1**	**2**	**≥3**
No. of affected probe sets	18,128	3,749	776	157

Table 8.3: The numbers of probe sets in Arabidopsis that have particular numbers of G-stack probes

It is noted that the percentage of probe sets containing probes with two or more G-stacks

in the Arabidopsis GeneChip is 4%. The actual number of probes with 0, 1, 2, or 3 or more G-stacks is shown in Table 8.3. From Table 3.1 it is seen that the percentage of probe sets containing probes with two or more G-stacks in the HG_U133A GeneChip is 14%. The similar percentage for affected probe sets in the HG_U133_Plus2 GeneChip is 11%. The greater percentage of probe sets containing probes with two or more G-stacks in the human GeneChips compared to Arabidopsis is an explanation for Memon's report and also explains the findings of this wide scale study that G-stacks are causing less bias in experiments that use Arabidopsis GeneChips than in those which use human GeneChips.

8.5 Comparison of Pseudomonas with Arabidopsis and Human microarray data

8.5.1 Pseudomonas expression data

Pseudomonas expression data is considered first by comparing graphs whose data has been processed with the RMA and PLIER normalization routines respectively. Figure 8.10 shows on the left, that using RMA the median differences between omitting probes with 2 or more G-stacks and leaving them in, are spread around the origin though not as tightly grouped at the origin as the control data for C-stacks. In the case of using PLIER on the right, the data for G-stacks shows that PLIER may have over-compensated for the presence of G-stacks because the median differences are strongly positive compared to the C-stack control.

The Pseudomonas expression data was then compared to that of the other two species: two human chips and Arabidopsis. It can be seen in Figure 8.11 that the Pseudomonas data in the right hand white columns of all the grouped bars is tending to the positive side of the origin, more than any of the other species. This shows that the G-stack bias seen by using RMA on microarray data from the human GeneChips is not evident in using Pseudomonas chips.

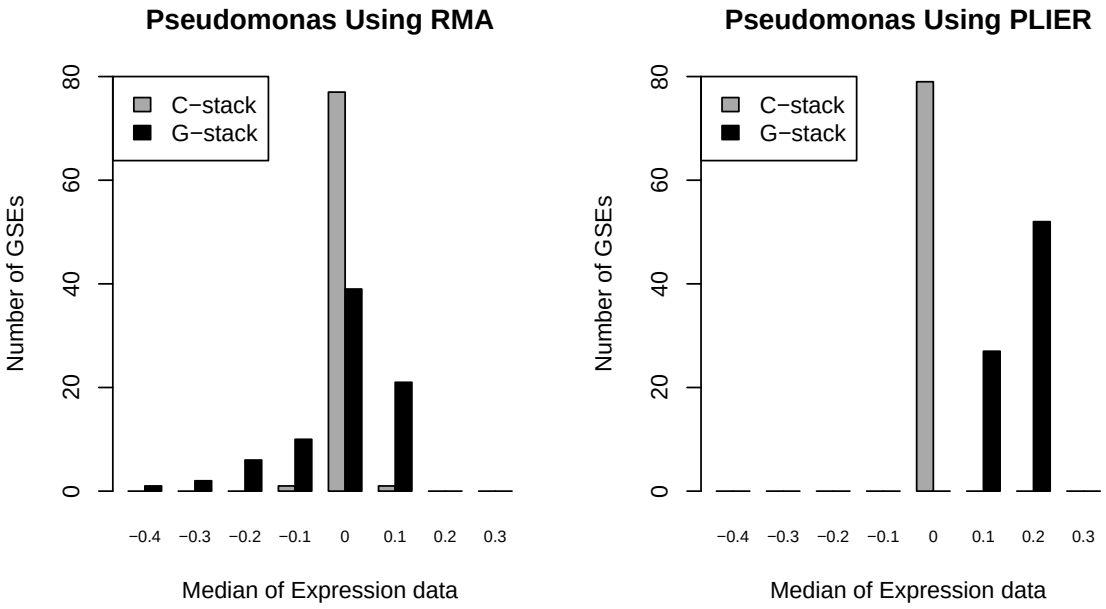


Figure 8.10: Expression data for Pseudomonas GeneChip

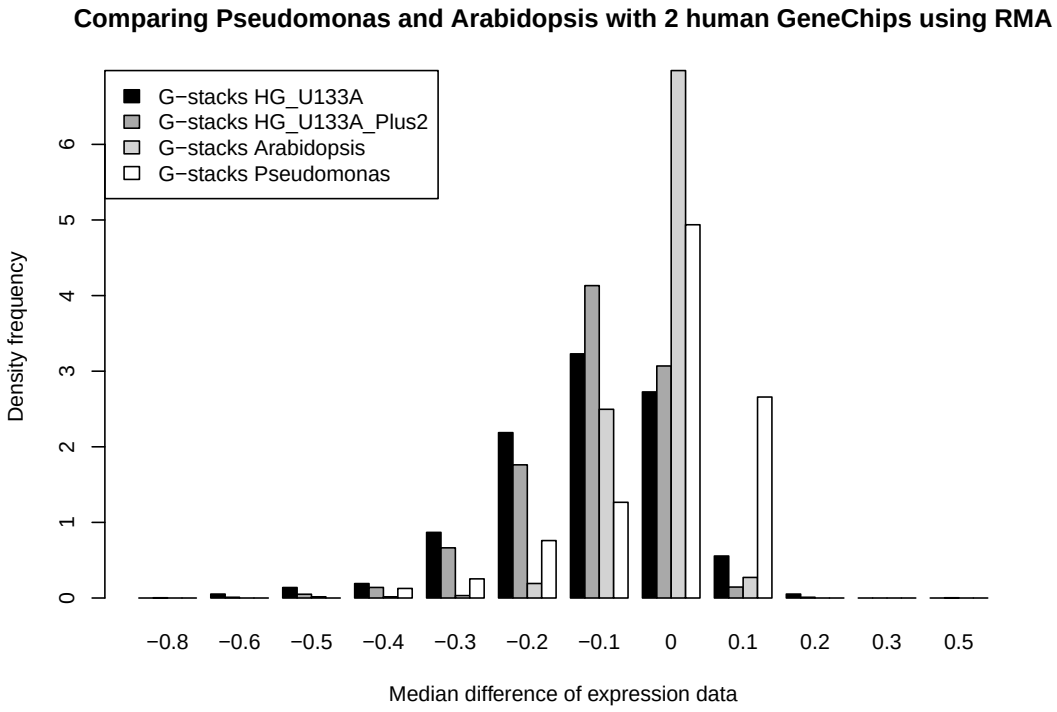


Figure 8.11: Comparing G-stacks in Pseudomonas and Arabidopsis GeneChips with those in HGU_133A and HGU_133_Plus2 GeneChips

8.5.2 Pseudomonas GT correlation data

It is noted that the work of Memon in her PhD study [70] demonstrated that PLIER showed a smaller bias than other normalization routines, particularly in correlations. From Figure 8.10 which shows expression data, it has been noted that PLIER tends to over compensate for the G-stack bias. In Figure 8.12 the median differences of GT correlation data are shown for RMA and for PLIER normalizations. In the right hand graph, PLIER shows no bias for G-stacks in this case, whereas in the left hand graph, the G-stack spread shows some bias compared the the control of C-stacks.

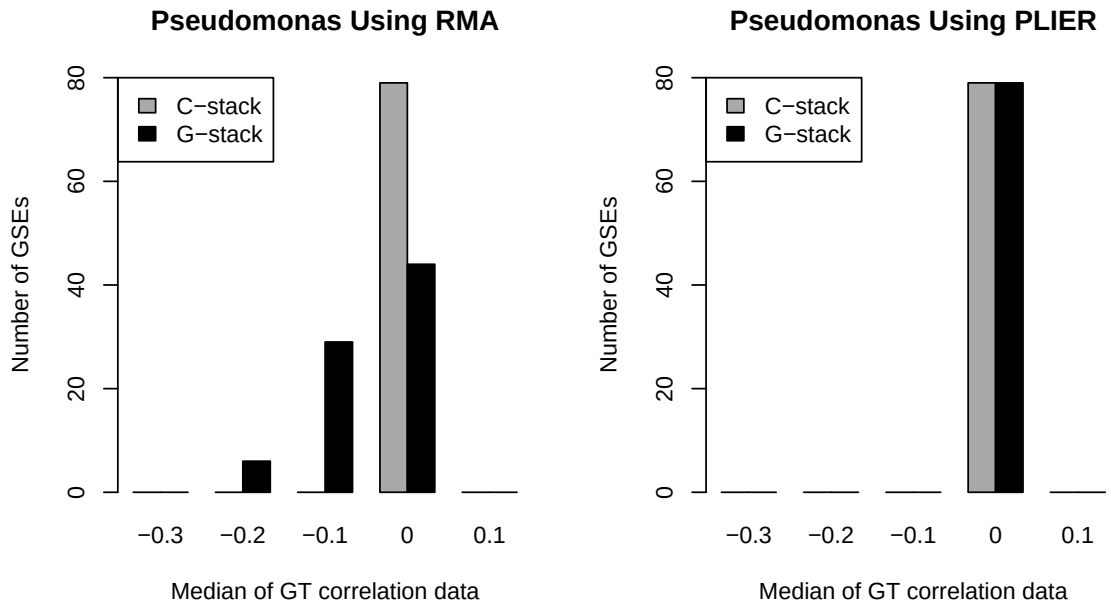


Figure 8.12: Comparing median difference in GT correlation data using RMA and using PLIER

In the previous section the comparison of expression data across four species has already been made and conclusions drawn. Here the comparison is made of GT correlation data across four species. In Figure 8.13 the side-by-side barplots are shown for both RMA and PLIER of the GT correlation data. The RMA case shows that the G-stack bias is most pronounced in the two human chips. The Arabidopsis chip shows some bias as the histograms are still predominantly negative, and the Pseudomonas chip has mild evidence of negative bias.

Using PLIER Figure 8.13 shows no G-stack bias for Pseudomonas, very little for Arabidopsis and small amounts for the two human chips.

8.6 Summary of wide scale analysis results

The use of scatter plots in section 8.2 showed clearly that in all four types of GeneChip surveyed the use of the PLIER normalization pipeline concentrated gene expression data in tighter bounds. In doing so, the PLIER algorithm compensated for G-stacks to a greater extent than did the RMA algorithm.

There were outliers found in the results when using RMA which were difficult to explain. Having looked at the metadata available for each outlier experiment, there was no apparent reason for the outliers in the subject or type of the experiments. Indeed if wet lab conditions or procedures played a part in producing these outliers, they were not apparent when the same data were processed by the PLIER algorithm.

The comparison of the two types of human GeneChips in section 8.3 showed that expression data from both chips showed differences when processed using the PLIER normalization pipeline rather than using the RMA normalization pipeline. For both chips PLIER had overcompensated for G-stacks though C-stacks were as expected as a control.

When comparing the processing of G-stacks directly, Figure 8.6 showed that the GeneChip HG_U133_Plus2 expression data is slightly less biased than that from HG_U133A (both were processed using RMA). The HG_U133_Plus2 chip is a later design with fewer G-stacks in its probes, so this result is to be expected. The GT correlation data is also less biased by G-stacks when processed with PLIER, as seen in Figure 8.7.

The comparison of Arabidopsis and Human GeneChip expression data showed that Arabidopsis data suffer less from G-stack bias than either set of the human chip data. A survey of the number of probe sets of the Arabidopsis chip which contain G-stacks was shown to be fewer than those in the two human chips. This is commensurate with the results

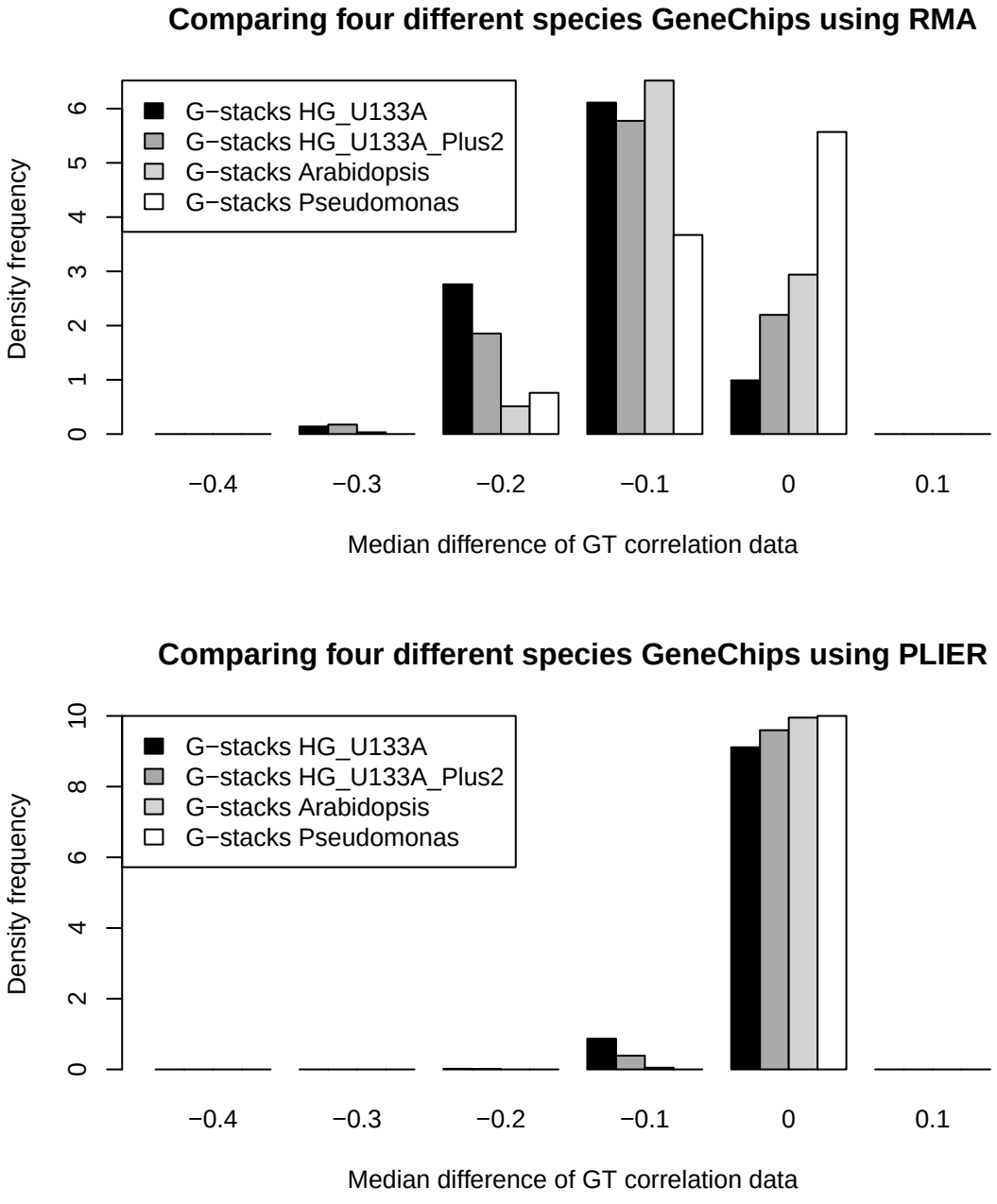


Figure 8.13: Comparing median difference in GT correlation data for four species using RMA and using PLIER

discovered.

Pseudomonas and Arabidopsis chip expression data both suffered from over compensation when using the PLIER normalization pipeline. However they both showed less bias in their expression data when compared to the human chip data, using RMA, (Figure 8.11). This is probably because of the fewer G-stack probes chosen for both the Pseudomonas and Arabidopsis microarrays.

Finally, the GT correlation data comparison of all four chips shows that PLIER eliminates G-stack bias more effectively than RMA for all the chips. This is apparent in Figures 8.12 and 8.13. Figure 8.13 also shows again that Arabidopsis and Pseudomonas have less inherent G-stack bias than the human chips.

Chapter 9

Conclusions

This chapter summarises the conclusions that can be drawn from the two aspects of this research study: the bioinformatics research into microarray data and the use of different cloud computing platforms. The research which gives rise to this thesis has taken place over nearly six years. During this time the climate for academic research has changed in remarkable ways. The increase in use and familiarity with next generation sequencing has continued apace, and yet the comparison of the use of microarrays and the use of new sequencing techniques has not always proven that the new sequencing techniques are an improvement upon microarrays for accuracy of expression data. Willenbrock *et al.* [71], for example, have found "that microarray expression measures actually correlate better with sample RNA content than expression measures obtained from sequencing data". They also report that microarrays perform with similar results to next-generation sequencing when it comes to reproducibility and relative ratio quantification.

It is important to know when making general statements like that from Willenbrock *et al.*'s abstract above, which platform has been used for the sequencing and what type of samples were analysed. Here it is sufficient to say that researchers must strive to ensure that their results are reproducible and as unbiased as possible with regard to the technology employed.

The development and public availability of cloud platforms has transformed the research

environment over the past six years. More clouds are available, with a wide variety of offerings in the area of utility computing. Pricing is now keenly competitive, with the possibility, for example, to choose cheaper options for computer runs if the time of day or immediacy of the experimental runs are not important.

9.1 Summary of cloud computing experiences

Grid computing and cloud computing were introduced in chapter 2. Experience of using a grid and of using clouds has been detailed in chapters 4 to 6.

9.1.1 Grid computing

It was found that grid computing was not easy to access in the case of the NGS. Each node had its own software with varying system implementation levels and procedures. There were restrictions on the length of jobs that could be run and therefore on the size of datasets that could be processed. It was not easy to transfer data files or software between nodes, so that jobs could not be easily transferred between nodes to take advantage of those nodes which were more lightly loaded. It was felt that the sharing of machines between a network of academic research centres was a good objective in principle, but difficult to implement in practice with limited systems administrator resources to facilitate the sharing.

9.1.2 Using Amazon EC2

This project used Amazon Web Services, including EC2 and Elastic Block Storage, from early days in the Amazon cloud offerings. The experience was good in that entry to use Amazon's computing services was gained very smoothly as soon as registration with a credit card had been successful. It was necessary to negotiate the security procedures and establish authorization to access storage areas and machine images. New machine images were created with the required Linux systems software, R and a few Bioconductor packages.

Programs in R were run to analyse some of the public datasets of the Ensembl Anotated Human Genome Data stored in Amazon Block Store. This work which identified the impact of G-quadruplexes on Affymetrix $3'$ arrays was published by Memon *et al.* [57].

The main conclusions from the Amazon experiences were that tight control must be exercised on the use of computing and storage resources. All machine instances which were launched were charged at a certain rate for complete hours, whether actually running any programs or not. A virtual machine running for 61 minutes would cost the price charged for two whole hours. This practice has since changed so that usage is now charged by the minute, and cloud computing pricing has become quite competitive as more companies have entered the field.

Storage facilities were quoted at a price per GigaByte per month. It was important not to waste money by storing unnecessary files on the cloud. Set against this were the advantages of cloud computing in being able to start up as many virtual machines as required for an analysis, and not have the overhead of buying physical machines, storing them, cooling them and maintaining them. The advantages of utility computing in only paying for what one uses are legion, and since the early days of Amazon the cloud computing market has expanded its offerings in many directions.

9.1.3 Use of a private cloud

The collaboration of this research project with another for which a private cloud was built has enabled insights into cloud computing research which would not have been possible as a user of public cloud services. The development of a virtual container as a service (vCAAS) has contributed a framework for optimising services in future cloud computing offerings. In this case the comparison of various queueing models was enabled in the situation where the same analysis programs were used on datasets which varied substantially in size.

Results were obtained which showed the advantage of taking into account the sizes of the datasets required for each job, and allocating jobs to particular queues which had the

appropriate bandwidth for downloading data from storage. Three different job scheduling algorithms were compared and the improvements in thrashing values determined by using the variant "with Lookup" were shown. In summary it was shown that the concept of vCAAS and its holistic view of resources was able to improve the performance of algorithms for resource provisioning and job allocation.

9.1.4 Using Windows Azure

Windows Azure was used for this project when Azure was at an early stage of usage by the Venus-C pilots. Windows programming and Web page interfaces were the only means of using Azure at that time. The experienced gained of web application development through .NET services using C# was invaluable. Web roles and worker roles were being developed by Microsoft alongside the Venus-C pilot developments.

The initial stage of developing a web page interface to run an R script was challenging. Then an R script was used to read microarray datasets from a public repository and upload them to Azure mass storage. Many challenges of data inconsistencies were overcome to arrange thousands of CEL files in their experiment folders ready for analysis.

Finally the analysis of the datasets was carried out, with results being returned initially to Azure storage, and then being downloaded to a local computer. During this phase the Generic Worker was used to streamline the submission of jobs to the Azure cloud, so that each job was run on a new VM. This method was superior to that in which a list of the datasets to be processed was submitted to each VM. In that case it often happened that a dataset would cause a job to fail because of data inconsistency problems or insufficient storage in the VM.

Windows Azure was a good cloud to use for Windows based work and for implementing a web based application. Its web roles and worker roles have been developed for this type of application. Although this research work did not test Azure's security and accounting features, it is understood that a variety of such services are offered.

9.2 Summary of microarray research

This work has shown that publicly available microarray data can be usefully reanalysed on a wide scale to determine the extent of possible bias due to technical reasons. The effect of having G-stacks in probes has been examined in four different GeneChips which were designed for three different species: human, a plant (Arabidopsis thaliana) and a bacterium (Pseudomonas aeruginosa).

The work of Shanahan *et al.* [38] has been confirmed in showing the significant bias due to G-stacks in the human GeneChip HG_U133A using a limited number of experiments. This work has broadened the results of Shanahan *et al.* with a wide scale survey of all available experiments of this GeneChip. It has been shown that probe sets containing G-stack probes are correlated with each other, whereas C-stack probes (acting as a control) are not correlated with each other (section 7.3.3).

The question of whether the number of CEL files in an experiment influenced either the expression data produced, or the GT correlation data, was investigated. It was found that the number of CEL files in an experiment did not affect the expression data or the GT correlation data results. However, small numbers of CEL files, i.e. one or two, should be avoided for statistical reasons.

The RMA and PLIER methods of normalization were compared because there are mentions in the literature of one or other method being superior. It was found that the PLIER method took longer to run and that the two methods did not give the same results either for summarized expression data or for GT correlation values. It was demonstrated that for HG_U133A data the PLIER method achieves improvement over the RMA method for ameliorating the bias caused by G-stacks in GT correlations (section 7.5.2). It was also shown that for this chip the gene expression data was more biased when using RMA than when using PLIER for normalization. Using a scatter plot, it was shown that when median shifts in expression data (between omitting and retaining G-stack and C-stacks) from all 576 available datasets were plotted the RMA method had a greater spread of the points than

the PLIER method. This tendency was apparent through similar scatter plots for each of the other species too.

The next GeneChip to be investigated with a wide scale survey was the human chip HG_U133_Plus2. The scatter plot for 1999 datasets of median shifts in expression data for this chip showed that outliers could be a significant feature. The specific outlier experiments were tracked down and reasons sought for their results. These experiments used a range of tissue types and the only common feature was the use of a small number of samples and therefore CEL files, typically only two or four. The experiment that had the greatest median shift in expression data for G-stacks using RMA was also an outlier when using PLIER, so this data produced consistent results with the two normalization methods.

The scatter plots for arabidopsis and for pseudomonas showed less spread and fewer noticeable outliers, though these outliers were again followed up. One experiment was again an outlier with both RMA and PLIER methods.

The microarray data from the two human GeneChips were compared with each other using the density frequency instead of the number of experiments. The comparison of expression data showed that the HG_U133_Plus2 chip was slightly less biased than the HG_U133A chip due to G-stacks and it was thought that this is probably due to it being a newer GeneChip design and having fewer G-stack probes. The percentage of probe sets containing probes with two or more G-stacks is 14% for the HG_U133A chip and 11% for the HG_U133_Plus2 chip.

The GeneChip for arabidopsis was designed later than both the human GeneChips considered in this work, and has fewer probe sets containing probes with 2 or more G-stacks. The percentage of such probe sets is 4%. When all available data for arabidopsis microarrays was analysed, it was found that the bias in expression data due to G-stacks using RMA was smaller than for both of the two human GeneChips. When using PLIER, it was found that an apparent overcompensation for G-stacks was evident, producing a bias in the other direction. The GT correlation data, however, was slightly biased by the G-stacks compared

with C-stacks using RMA, and even less biased by the G-stacks using PLIER.

Finally, the comparison of all available pseudomonas microarray data with that of the other three chips shows that using RMA, the gene expression data for pseudomonas has negligible bias. Using PLIER the gene expression data for pseudomonas is biased (like data for arabidopsis) in the opposite direction from that expected for G-stacks. This appears to be an over compensation by PLIER for G-stacks. The GT correlation data for pseudomonas is completely unbiased using PLIER, and very slightly biased using RMA.

9.3 Discussion

Cloud computing has become increasingly acknowledged as an important resource for bioinformatics research over the period of this work. The experience gained on the NGS, on Amazon EC2, on a private cloud and on Windows Azure serves to demonstrate part of the wide variety of offerings which are now available to help in the analysis of big data in the field of bioinformatics. In particular, in the area of surveying public microarray data, it has been shown that useful results can be obtained by storing a large amount of microarray data on a cloud and processing it with tools such as R and Bioconductor modules.

Significant results have been obtained from the wide scale survey of microarray datasets from four different species: two human, one plant and one bacterium. It was found that G-stacks cause a bias in 19% of datasets of one human GeneChip. The second and newer human GeneChip was found to have slightly less bias due to G-stacks when considering the gene expression data. The plant GeneChip *arabidopsis thaliana* was even less affected by G-stacks in its probes, and the bacterium GeneChip *pseudomonas aeruginosa* was affected the least. It was found that the normalization routine, PLIER, was in many cases better than RMA in ameliorating the effect of G-stacks, but that for processing expression data in the cases of arabidopsis and pseudomonas, PLIER overcompensated for the G-stack effect.

9.4 Future Work

As far as cloud computing is concerned, a useful development in this field would be for public databases to be sited in the same locations as cloud computing facilities. For example, if the EMBL-EBI at Hinxton offered not only a plethora of data from life science experiments, but also the cloud computing services to analyse the data, then many more wide scale surveys could be conducted. These could analyse high-throughput sequence data as well as microarray data, without the time-consuming and expensive necessity of downloading vast quantities of data to local machines.

In the area of microarrays it would be worthwhile to perform further analysis on the effect of other particular motifs encountered in sequences in microarrays as well as G-stacks, for example 'TTTTT', 'CCGCC' or 'CCTCC', as these were highlighted by Upton [4] as showing higher than expected cross-correlations. Another development could be the attempted correction of bias in microarrays due to certain sequences in probes.

There could be scope in the field of high-throughput sequencing for the checking and correcting of bias due to technical reasons. It is important for biologists to analyse several samples of their tissues so that consistent results can be verified. Further understanding of the strengths and weaknesses of the technology could benefit future research.

Appendices

Appendix A

CEL File Format

A.1 Version 3 Format

Version 3 format is also known as the ASCII version according to Affymetrix [72], and is generated by the MAS software. The format of the CEL file is an ASCII text file similar to the Windows INI format. The file is divided up into sections. The start of each section is defined by a line containing a section name enclosed in square braces. The section names are: “CEL”, “HEADER”, “INTENSITY”, “MASKS”, “OUTLIERS” and “MODIFIED”. The data in each section is of the format TAG=VALUE.

The **“CEL”** section contains the version number of the file. The TAGS are:

TAG	**Description**
Version	The version number. Always set to 3.

The **"HEADER"** section contains miscellaneous header information. The TAGS are:

TAG	Description
Cols	The number of columns in the array (of cells).
Rows	The number of rows in the array (of cells).
TotalX	Same as Cols.
TotalY	Same as Rows.
OffsetX	Not used, always 0.
OffsetY	Not used, always 0.
GridCornerUL	XY coordinates of the upper left grid corner in pixel coordinates.
GridCornerUR	XY coordinates of the upper right grid corner in pixel coordinates.
GridCornerLR	XY coordinates of the lower right grid corner in pixel coordinates.
GridCornerLL	XY coordinates of the lower left grid corner in pixel coordinates.
Axis-invertX	Not used, always 0.
AxisInvertY	Not used, always 0.
swapXY	Not used, always 0.
DatHeader	The header from the DAT file.
Algorithm	The algorithm name used to create the CEL file.
AlgorithmParameters	The parameters used by the algorithm. The format is TAG:VALUE pairs separated by semi-colons or TAG=VALUE pairs separated by spaces.

The "INTENSITY" section contains intensity information. The TAGS are:

TAG	Description
NumberCells	The total number of cells in the array (Rows*Cols)
CellHeader	The header for the remainder of the data in this section. The header is always set to: "X Y MEAN STDV NPIXELS"
NA	The remaining lines in this section contain the intensity, standard deviation value and the number of pixels used to compute the intensity value for each cell in the array. The order is defined by the header.

The "MASKS" section specifies which cells have been masked by the user. The TAGS are:

TAG	Description
NumberCells	The number of masked cells.
CellHeader	The header for the remainder of the data in this section. The header is always set to: "X Y".
NA	The remaining lines in this section contain the XY coordinates of those cells masked by the user.

The "OUTLIERS" section specifies which cells were called outliers by the software. The TAGS are:

TAG	Description
NumberCells	The number of outlier cells.
CellHeader	The header for the remainder of the data in this section. The header is always set to: "X Y".
NA	The remaining lines in this section contain the XY coordinates of those cells called outliers by the software.

The "MODIFIED" section specifies which cells were modified by the user. This feature was dropped in MAS 4 thus the number of cells in this section should always be 0. The TAGS are:

TAG	Description
NumberCells	The number of outlier cells.
CellHeader	The header for the remainder of the data in this section. The header is always set to: "X Y ORIGMEAN".
NA	The remaining lines in this section contain the XY coordinates and the original intensity value (calculated by the software) of those cells modified by the user.

A.2 Version 4 Format

Version 4 format of the CEL file is a binary file where values are stored in little-endian format. Big-endian and little-endian are terms which describe the order in which a sequence

of bytes are stored in computer memory. Little-endian is an order which first stores the "little end" (the least significant value in the sequence).

The file contents are defined by:

Table A.1: File Contents for Version 4 Format CEL files

Item	Description	Type
1	Magic number. Always set to 64.	integer
2	Version number. Always set to 4.	integer
3	Number of columns.	integer
4	Number of rows.	integer
5	Number of cells (rows*cols).	integer
6	Header length	integer
7	Header as defined in the HEADER section of the version 3 CEL files. The string contains TAG=VALUE separated by a space where the TAG names are defined in the version 3 HEADER section.	char [length defined above]
8	Algorithm name length.	integer
9	The algorithm name used to create the CEL file.	char [length defined above]
10	Algorithm parameters length.	integer
11	The parameters used by the algorithm. The format is TAG:VALUE pairs separated by semi-colons or TAG=VALUE pairs separated by spaces.	char [length defined above]
12	Cell margin used for computing the cell's intensity value.	integer
13	Number of outlier cells.	DWORD

Continued on next page

Table A.1 – continued from previous page

Item	Description	Type
14	Number of masked cells.	DWORD
15	Number of sub-grids.	integer
16	Cell entries - this consists of an intensity value, standard deviation value and pixel count for each cell in the array. The values are stored by row then column starting with the X=0, Y=0 cell. As an example, the first five entries are for cells defined by XY coordinates: (0,0), (1,0), (2,0), (3,0), (4,0).¡ /p¿	(float, float, short)
17	Masked entries - this consists of the XY coordinates of those cells masked by the user.	(short, short)
18	Outlier entries - this consists of the XY coordinates of those cells called outliers by the software.	(short, short)

Continued on next page

Table A.1 – continued from previous page

Item	Description	Type
19	Sub-grid entries - This is the sub-grid definition. There are as many sub-grids in the file as defined by the number of sub-grids above. Each sub-grid is defined as: - row number (integer) - column number (integer) - upper left x coordinate in pixels (float) - upper left y coordinate in pixels (float) - upper right x coordinate in pixels (float) - upper right x coordinate in pixels (float) - lower left x coordinate in pixels (float) - lower left y coordinate in pixels (float) - lower right x coordinate in pixels (float) - lower right x coordinate in pixels (float) - left cell position (integer) - top cell position (integer) - right cell position (integer) - bottom cell position (integer)	(integer, integer, float, float, float, float, float, float, float, integer , integer , integer , integer)

Types used are defined as: integer (32-bit signed integer), DWORD (32-bit unsigned integer), float (32-bit floating-point number), short (16-bit signed integer).

Appendix B

Chip Definition Files

B.1 Tables describing each section of CDF files

The **"CDF"** section contains the version number of the file. The TAGS are:

TAG	Description
Version	The version number. Should always be set to "GC1.0", "GC2.0", "GC3.0", "GC4.0", "GC5.0", or "GC6.0". This document describes GC3.0, GC4.0, GC5.0, and GC6.0 version CDF files.
GUID	The unique identifier of the CDF. (Only available in version 6)
md5	The integrity md5 of the CDF. (Only available in version 6)

The **"Chip"** section contains the following TAGS:

TAG	Description
Name	The name of the array. This item is not used by the software.
ChipType	The probe array type. Multiple entries may exist. (Only available in version 6)
Rows	The number of rows of cells on the array.
Cols	The number of columns of cells on the array.
NumberOfUnits	The number of units in the array not including QC units. For CustomSeq arrays, there are 2 units: Unit1 contains the probes interrogating a sense target and Unit2 contains the probes interrogating an anti-sense target. For all other array types, there exists one unit per probe set.
MaxUnit	Each unit is given a unique number. This value is the maximum of the unit numbers of all the units in the array (not including QC units).
NumQCUnits	The number of QC units. QC units are defined in version 2 and above. CustomSeq arrays do not contain any QC units.
ChipReference	Used for CustomSeq, HIV and P53 arrays only. This is the reference sequence displayed by the Affymetrix software. The sequence may contain spaces. This value is defined for version 2 and above.

The next set of sections where the name begins with "QC" define the QC units or probe sets in the array. There are NumQCUnits (from the Chip section) QC sections.

Each section name is a combination of "QC" and an index ranging from 1 to NumQCUnits-1 and will be listed sequentially. QC units are defined for version 2 and above.

Each section contains the following TAGS:

TAG	Description
Type	Defines the type of QC probe set. The defined types are: 0 - Unknown 1 - Checkerboard Negative 2 - Checkerboard Positive 3 - Hybridization Negative 4 - Hybridization Positive 5 - Text Features Negative 6 - Text Features Positive 7 - Central Negative 8 - Central Positive 9 - Gene Expression Negative 10 - Gene Expression Positive 11 - Cycle Fidelity Negative 12 - Cycle Fidelity Positive 13 - Central Cross Negative 14 - Central Cross Positive 15 Cross Hyb Negative 16 Cross Hyb Positive
NumberCells	The number of cells in the probe set.
CellHeader	Defines the data contained in the subsequent lines, separated by tabs. For all QC probe set types: X - The X coordinate of the cell. Y - The Y coordinate of the cell. PROBE - The probe sequence of the cell. Typically set to "N". PLEN - The number of bases in the probe sequence. ATOM - An index used to group multiple cells. INDEX - An index used to look up the corresponding cell data in the CEL file. The final data items are dependent on the type of the QC probe set: MATCH - A boolean flag indicating a perfect match probe. For types: 7 - Central Negative, 8 - Central Positive, 9 - Gene Expression Negative, 10 - Gene Expression Positive BG - A boolean flag indicating a background (blank) cell. For types: 9 - Gene Expression Negative, 10 - Gene Expression Positive CYCLES - This item is always a list of 0's separated by a tab. There are as many 0's as number of bases in the probe sequence (PLEN). For types: 11 - Cycle Fidelity Negative, 12 - Cycle Fidelity Positive
Cell*i*	This contains the information about a cell that belongs to the probe set. The value of i in the tag ranges from 1 to the number of cells in the probe set and will be listed sequentially. The values in each line depend on the CellHeader. The values are separated by tabs.

The next set of sections where the name begins with "Unit" define the probes that are a member of the unit (probe set). Each unit is divided into subsections termed "Blocks" which are referred to as "groups" in the Files SDK documentation.

Each section name is a combination of "Unit" and an index. There is no meaning to the index value. Immediately following the "Unit" section there will be the "Block" sections

for that unit before the next unit is defined.

Each “Unit” section contains the following TAGS:

TAG	Description
Name	The name of the unit. The probe set name for Genotyping, Copy Number, Polymorphic Marker and Multichannel Marker units or “NONE” for all other unit types.
Direction	Defines if the probes are interrogating a sense target or anti-sense target (1 - sense, 2 - anti-sense, 3 - both).
NumAtoms	The number of atoms in the entire probe set. This TAG name contain two values after the equal sign. The first is the number of atoms and the second (if found) is the number of cells in each atom. An atom is a probe quartet for CustomSeq units and a probe pair for all other unit types.
NumCells	The number of cells in the entire probe set. Probe pairs contain 2 cells and probe quartets contain 4 cells.
UnitNumber	An arbitrary index value for the probe set.
UnitType	Defines the type of unit (0 - Unknown, 1 - CustomSeq, 2 - Genotyping, 3 - Expression, 7 - Tag/GenFlex, 8 - Copy Number, 9 - Genotyping Control, 10 - Expression Control, 11 - Polymorphic Marker, 12 - Multichannel Marker). An array may contain units of varying types.
NumberBlocks	The number of blocks or groups in the probe set.
MutationType	Used for Genotyping units only in defining the type of polymorphism (0 - substitution, 1 - insertion, 2 - deletion). This value is available in version 2 and above.

After the “Unit” section follows the “Unit-Block” sections. There are as many “Unit-Block” sections as defined by NumberBlocks. A block will list the probes as its members.

The TAGS are:

Table B.1: File Contents for CDF files

Tag	Description
Name	The name of the block. For Genotyping units this is the allele. For Polymorphic Marker and Multichannel Marker units this is “None”. For all other unit types this is the name of the probe set.

Continued on next page

Table B.1 – continued from previous page

Tag	Description
BlockNumber	An index to the block.
Wobble	The wobble situation for Polymorphic Marker and Multichannel Marker units in the block. Only available in version 4, 5, and 6.
Allele	The allele code for Polymorphic Marker and Multichannel Marker units in the block. Only available in version 4, 5, and 6.
Channel	The channel code for multichannel microarray platform. Only available in version 5 and 6.
RepType	The probe replication type (0 - unknown, 1 - different probe sequences, 2 - some probe sequences are identical, 3 - all probe sequences are identical) for probe set groups used under multichannel microarray platform. Only available in version 5 and 6.
NumAtoms	The number of atoms in the block.
NumCells	The number of cells in the block.
StartPosition	The position of the first atom.
StopPosition	The position of the last atom.
Direction	Used for Genotyping, Polymorphic Marker and Multichannel Marker units only in defining whether the probes are interrogating a sense target or anti-sense target (0 - no direction, 1 - sense, 2 - anti-sense). This value is available in version 3 and above.

Continued on next page

Table B.1 – continued from previous page

Tag	Description
CellHeader	Defines the data contained in the subsequent lines, separated by tabs. The values are: X- The X coordinate of the cell. Y - The Y coordinate of the cell. PROBE- The probe sequence of the cell. Typically set to "N". FEAT - Unused string. QUAL - The probe set name plus the allele for Genotyping units. The probe set name for all other unit types. EXPOS - Ranges from 0 to the NumAtoms - 1 for Expression units. For all other unit types, provides relative positional information for the probe. PLEN - The length of probe sequence. Only available in version 4, 5, and 6. POS - An index to the base position within the probe where the mismatch occurs. CBASE - Not used. PBASE - The probe base at the substitution position. TBASE - The base of the target where the probe interrogates at the substitution position. ATOM - An index used to group probe pairs or quartets. For Expression, identical to EXPOS. INDEX - An index used to look up the corresponding cell data in the CEL file. GROUP - The physical grouping of probe on the array. Only available in version 4, 5, and 6. The following are only available in version 2 and above: CODONIND - Always set to -1 CODON -Always set to -1 REGIONTYPE - Always set to 99 REGION - Always set to a blank character
Cell*i*	This contains the information about a cell that belongs to the block. The value of i in the tag ranges from 1 to the number of cells in the block. The values in each line depend on the CellHeader. The values are separated by tabs.

XDA Format

The format of this CDF file is a binary file created for faster access and smaller file size. The values in the file are stored in little-endian format.

The file contents are defined by:

Table B.2: File Contents for XDA Format CDF files

Item	Description	Type
1	Magic number. Always set to 67.	integer
2	Version number. Should set to 1, 2, 3, or 4.	integer
3	The length of the GUID, an unique identifier of the CDF. (Only available in version 4)	unsigned integer
4	GUID, the unique identifier of the CDF. (Only available in version 4)	char [length defined above]
5	The integrity md5 of the CDF. (Only available in version 4)	char[32]
6	The number of probe array types. (Only available in version 4)	unsigned char
7	The length of probe array type. (Only available in version 4)	unsigned integer
8	The probe array type. (Only available in version 4)	char [length defined above]
9	The length and value of probe array type as described in Item 7 and 8 respectively if there is more than one entry. (Only available in version 4)	(unsigned integer + char [length defined]) * (# of probe array types - 1)
10	The number of columns of cells on the array.	unsigned short
11	The number of rows of cells on the array.	unsigned short

Continued on next page

Table B.2 – continued from previous page

Item	Description	Type
12	The number of units in the array not including QC units. The term unit is an internal term which means probe set.	integer
13	The number of QC units.	integer
14	The length of the CustomSeq reference sequence.	integer
15	The CustomSeq reference sequence.	char [length defined above]
16	The probe set name. The UNIT name for CustomSeq, Genotyping, Polymorphic Marker, and Multichannel Marker. The BLOCK name for Expression.	char[64] * (# of units)
17	File position for the start of each QC unit information block.	integer * (# of QC units)
18	File position for the start of each unit information block.	integer * (# of units)
19	QC information, repeated for each QC unit: Type - unsigned short Number of probes - integer Probe information, repeated for each probe in the QC unit: X coordinate - unsigned short Y coordinate - unsigned short Probe length - unsigned char Perfect match flag - unsigned char Background probe flag - unsigned char	see description

Continued on next page

Table B.2 – continued from previous page

Item	Description	Type
20	Unit information, repeated for each unit: UnitType - unsigned short (1 - Expression, 2 - Genotyping, 3 - CustomSeq, 4 - Tag, 5 - Copy Number, 6 - Genotyping Control, 7 - Expression Control, 8 - Polymorphic Marker, 9 - Multichannel Marker) Direction - unsigned char Number of atoms - integer Number of blocks - integer (always 1 for Expression units) Number of cells - integer Unit number (probe set number) - integer Number of cells per atom - unsigned char Block information, repeated for each block in the unit: Number of atoms - integer Number of cells - integer Number of cells per atom - unsigned char. Direction - unsigned char. The position of the first atom - integer <unused integer value> - integer The block name - char[64]. Wobble situation - unsigned short (only available in version 2, 3, and 4). Allele code - unsigned short (only available in version 2, 3, and 4). Channel - unsigned char (only available in version 3 and 4). RepType - unsigned char (0 - unknown, 1 - different probe sequences, 2 - some probe sequences are identical, 3 - all probe sequences are identical) (Only available in version 3 and 4).	see description

Continued on next page

Table B.2 – continued from previous page

Item	Description	Type
20 cont.	Cell information, repeated for each cell in the block: Atom number - integer X coordinate - unsigned short Y coordinate - unsigned short Index position (relative to sequence for CustomSeq, Genotyping, Copy Number, Polymorphic Marker, and Multichannel Marker units, for Expression units this value is the atom number) - integer Base of probe at substitution position - char Base of target at interrogation position - char Length of probe sequence - unsigned short (only available in version 2, 3, and 4) Physical grouping of probe - unsigned short (only available in version 2, 3, and 4)	see description

Appendix C

Unique Mappings

This appendix give the detailed description of how unique mappings are obtained.

C.1 Required Software

Linux OS

mysql

perl

megablast

gawk

The programs that process the data and obtain the unique mappings are mainly written in perl and use mysql databases. A separate database exists for every species that is processed. For example, for Human data from Ensembl version 57, the database Human_Ensembl57 must exist. The program 'gawk' is used for some simple data selection with defined bounds, to reduce the amount of data being handled by the perl programs. This technique reduces the overall time taken to produce the unique mappings for any particular species.

C.2 Obtaining the Data

The first step is to obtain the data needed for the process of generating unique mappings. The data comprises:-

1. exon sequences
2. transcript sequences
3. probe sequences

The process of downloading each of these sequences will be described now.

1. Sequences of the Ensembl exons

   The FASTA files are obtained by using Biomart at **http://www.biomart.org/biomart/-martview/** or **http://www.ensembl.org/biomart/martview/** or for the plant species at **http://plants.ensembl.org/biomart/martview/**.

   We are currently using Ensembl Release 57. There are two files downloaded for the exons:-

   - Ensembl57Exons.txt contains Ensemble Exon IDs and Exon Nucleotide sequences. This file is used by *megablast* to get the slignments of probes to exons.
   - Ensembl57ExonsF.txt contains Exon IDs, Start Position, End Position, Gene ID, Transcript IDs, Chromosome, Strand and Exon Nucleotide sequences. This file is used by the perl programs in Section C.5.
   - Ensembl57SplicesTrans.txt contains Ensembl Transcript IDs and cDNA Nucleotide sequences. This file is used by *megablast* to get the alignments of probes to transcripts.
   - Ensembl57SplicedTransF.txt contains Gene ID, Chromosome, Transcript ID, Start Position, End Position, Strand, Exon IDs, Exon Rank in Transcript and

cDNA Nucleotide sequences. This file is used by the perl programs in Section C.5.

It is also possible to install the perl package called Biomart-perl, and write a perl program to download the data. However, when this was tried there were problems with the connection to Biomart which led to incomplete data being received. It was found preferable to use the Biomart webpage facility to obtain the data. Once the fields had been selected, and 'Results' chosen, the option 'Compressed web file (notify by email)' was used with the insertion of an email address to receive notification when the data was ready to download.

2. Probe sequences: the FASTA files are obtained from the Affymetrix web page

   http://www.affymetrix.com/products_services/index.affx. For exon arrays it is also necessary to download the .tab file. For example,

   HuEx_1_0_st_v2.probe.tab.

C.3 *Megablast* Alignments

Megablast is used to align the probe sequences to the Ensembl exon and transcript sequences. Details of the use of this program can be found at:- **http://blast.ncbi.nlm.nih.gov/Blast.cgi**. The version of *megablast* used for the current mappings is 2.2.17. The commands to generate the mappings are:-

- **formatdb** -i SequencesFastaFile -p F -o T
- **megablast** -W 10 -r 4 -q -5 -G 12 -E 8 -F F -s 20 -p 80 -d SequencesFastaFile -i ProbeSequenceFastaFile -D3 -o Mappings OutputFile

formatdb has to be run before the megablast command. This prepares the given file to be processed by *megablast*. It generates files with indexes to make the *megablast* processing faster.

The purpose of using the parameters shown for the *megablast* command is to obtain mappings in which in the worst case only 10 of the 25 bases of a probe align to an exon or a transcript.

C.3.1 Megablast parameters

The megablast parameters used above are described in detail here:-

- **-W 10** Word size. Default 28.
- **-r 4** Match score. This sets the score of a nucleotide match. See -q and scoring scheme appendix B in manual.
- **-q -5** Sets the penalty for a nucleotide mismatch. The choice of integer for -q and -r are very important because they determine your target frequencies.
- **-G 12** Initial penalty for opening a gap of length 0. Penalties for extending the gap are controlled by parameter -E.
- **-E 8** Setting -E and -G turns on affine gapping, so that there is a greater penalty for opening a gap than for extending the gap.
- **-F F** False. True filters a given sequence string. This had to be switched off to stop megablast from filtering out the low-complexity sequences which are quite repetitive.
- **-s 20** The minimum hit score to report. All alignments scoring less than this value are not reported.
- **-p 80** Percent identity cutoff. Alignments less than 80% are not reported.
- **-d** The database or file to be compared.
- **-i** The query file of probe sequences.

C.4 Configuration Files

There are two configuration files which hold data to direct and specify details to the perl programs. For each species, the following files must be available:-

- specieConfig.txt - This file contains information used by the perl programs. For example:-

  db=Ensembl

  release=57

  specie=Mouse

  prog_version=v7

  prog_status=p (p means production, d means development)

  source_dir=/home/owena

  output_dir=/cluster

  db_type=Ensembl

C.5 Auxiliary Data

For each new species and array, or new version of a species to be processed, it is necessary to populate the corresponding tables. The tables used are:-

- **EXONS** contains exon information, exon_id, initial_position, end_position, gene_id, transcript_list, chromosome, strand
- **TRANSCRIPTS** contains spliced transcript information: transcript_id, initial_position, end_position, gene_id, chromosome, strand, exon_list, exon_order
- **EXON_SEQ** contains exon sequences: exon_id, sequence

- **EXON_SYNONYMS** contains sysonymous exons: exon_id, exon_essex_id, chromosome, strand, gene_id. One exon_essex_id can have two or more exon_ids.

- **PROBES** contains probes information: probe_id, senseness, interrogation_position, sequence, order_number (position of the probe in the probeset). The control probes in $3'$ arrays are not considered. In exon arrays the information is: probe_id, probeset, x_position, y_position, chromosome, initial_position, end_position, strand, probe_sequence, category.

It should be noted that the name of the sequence files of robes for each array had to contain "_" instead of "-" in order to be read by the perl programs, and accepted by sql in the database. For example MoEx-1_0-st-v1.probe.tab had to be modified to MoEx_1_0_st_v1_probe.tab.

The database where the tables are to be located had to be created manually before running the perl programs. Certain directories must also exist before running the programs which will use them.

The following perl programs were used:-

- **populate_initial_tables** to create the required tables in the mysql database

- **exonsTransInf** to read the exons and transcripts information, populate those tables and create .csv files of the same data

- **getCSVProbes** to read the probes information, populate the probes table and create a .csv file

- **getProbesExonArrays** to read the probes information for exon arrays, populate the probes table and create a .csv file

- **exon_synonyms_from_chrom** to work out and obtain synonymous exons, populate the synonymous exon table and .csv file.

C.6 Obtain Probes uniquely mapping to exon-junctions

It was desirable to take out those probes which map to exon-junctions. To do this, there were four stages necessary to produce a table of such probes which map uniquely to exon-junctions.

C.6.1 Initial Filtering

The mappings files were filtered before being processed by the perl programs because they could be huge files. The filtering saved much time and resources. Both the exons and the spliced transcripts files were filtered using gawk, to restrict the alignment length to that with values greater than or equal to 20 bases, and with a minimum of 80% identity. These are example gawk filtering commands...

gawk '$3 >= 80 && $4 >= 20 {print $0}' MoEx_1_0_st_v1Exons.txt > MoEx_1_0_st_v1Exons-filtered1.txt

gawk '$3 >= 80 && $4 >= 20 {print $0}' MoEx_1_0_st_v1SplicedTrans.txt > MoEx_1_0_st_v1SplicedTrans-filtered1.txt

C.6.2 Creation of Exon mappings table

Using the filtered exons file, a perl program called **create_exon_map_tables** processed the records to populate a new table of Exon mappings. It also produced a .csv file of the same data.

C.6.3 Creation of Spliced Transcript mappings table

Using the filtered spliced transcript file, a perl program called **create_trans_map_tables** processed the records in a similar way to populate a new table of Spliced Transcript mappings, and created a .csv file.

C.6.4 Identifying probes which map to exon-junctions

The perl program called **getExonJunct** was used to identify the probes which map to exon junctions. First the program found the exons of a given transcript, and then read their initial position and length from the EXON table. If exons were in the junction, then their details were written to a new table and to a .csv file.

C.7 Unique Mappings

Having prepared all the required tables and files as described, it was now possible to execute the program to generate unique mappings to exons. The perl program called **uniqueExons-v7** was used to generate the files that contain the unique mappings in two formats: the ones used by the pipeline to generate heatmaps, and .csv files to populate the database. In particular the following files were created:-

- **list_ExonProbeUnique.txt** a list of probes uniquely mapping to exons
- **listAntisense.txt** a list of probes uniquely mapping to exons in the antisense direction (i.e. start position $>$ end position). These probes are a subset of **list_ExonProbe Unique.txt**.
- **listSense.txt** a list of probes uniquely mapping to exons in the sense direction (i.e. end position $>$ start position). These probes are a subset of **list_ExonProbeUnique.txt**.
- **listCombinedSense.txt**

The form of the data in these files is as shown by the following example:-

ENSE00000330966: 535192445-1524-2090:678-654 266715443-2192-1041:1601-1577

where the fields are:-

exon-id: probeId-x-y:startPos-endPos probeId-x-y:startPos-endPos

Finally, a perl program called **populateTables** was run to use the .csv file to create tables of unique mappings in the database.

Appendix D

Private Cloud Details

Much of the detail here is taken from the published paper by Musa *et al.* [61].

D.1 Infrastructure Setup

A small room was dedicated to the private cloud facility within the School of Computer Science and Electronic Engineering at the University of Essex. The following are the hardware components which were connected and deployed to create the private cloud:-

1. Ten homogeneous 64 bit Dell OptiPlex systems with 4GB RAM, core 2 Duo processor running at 3.33GHz, and 300GB hard disk storage capacity. Each computer had four virtual cores per processor. The internal prototype experiment cloud is therefore created on the 40 cores of total processing capacity and 80GB of total RAM. XEN Cloud Platform (XCP) was deployed on all the ten servers for hosting virtual machines. Deploying XCP enabled the combination of all the machines into one large pool of memory and processing capacity.

2. One 64 bit Dell OptiPlex system with 8GB RAM and core 2 Duo processor running at 3.33GHz was set aside for the Network File System (NFS) server (NFSS). The computer had a total usable storage capacity of 20TB comprising the internal hard

disk and the external storage array directly attached to the server.

3. One 64 bit Dell OptiPlex system with 8GB RAM, core 2 Duo processor running at 3.33GHz and four cores, and a hard disk storage capacity of 300GB set aside for the management server (MS) which also serves as the Middleware server.

4. One 64 bit Dell OptiPlex system with 8GB RAM, core 2 Duo processor running at 3.33GHz was set aside for storing results of gene expression in a MySQL database server (DBS). The DBS has a capacity of 300GB Hard disk and four virtual cores per processor.

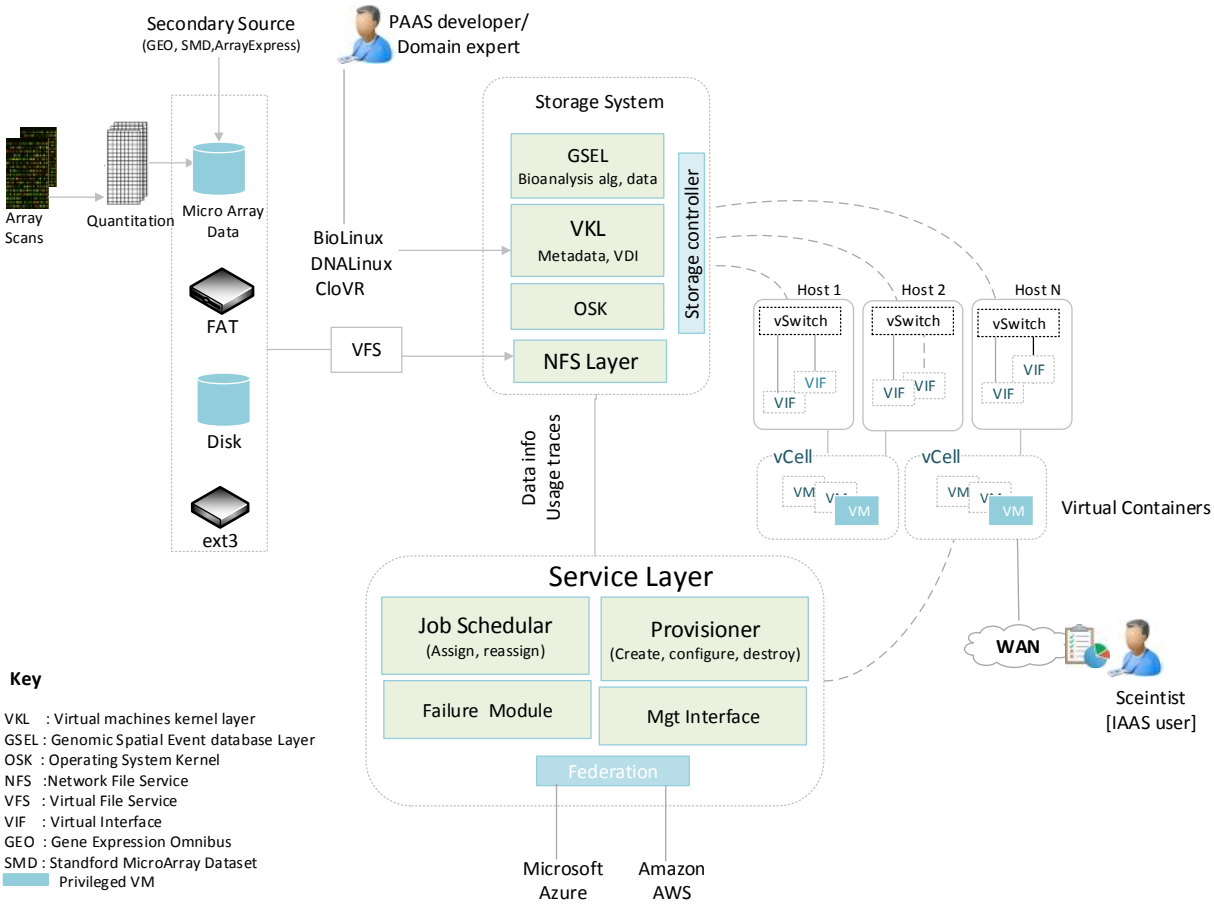


Figure D.1: Architecture of resource interactions in the private cloud

Figure D.1 shows how the private cloud was configured to provide a test bed for the microarray data analysis application. The following features should be noted:-

- Inexpensive second hand machines with network switches were deployed.
- Physical hosts were connected in a three-layered hierarchical network topology (Figure D.1) comprising aggregation, edge and access layers.
- Basic VM modules were created and packaged in a privileged virtual machine.
- The privileged VM (blue VMs in Figure D.1) is created and configured with functionalities for resource provisioning and dynamic job allocation. This packaged pVM was created in the requested container.
- Each virtual service cell is equipped with basic service functionalities suitable for the microarray data analysis.
- Experiment data stored in various storage media were made available via a standard network file system.
- The functionalities available in the vCell estimated finish time based on previous time series statistics of finished job and dynamically adjusted the operations of various components.
- Provisioning and job allocation algorithms were tested using the value K=3 for the job queues.

D.1.1 vCell Implementation

The code to realize the virtual infrastructure container was divided into five sets of modules. Each module was created to perform an interdependent task. The modules were for vCell creation, vCell request creation, storage access, gene expression identification, and the result writer. The functions of provisioning, allocation and report writing are projected into the pVM. The provisioning is invoked by the pVM which starts the operation of the provisioner.

The vCell resources (VM, VIF) creation modules reside in pVM enabled with a subset of the service layer. Resource provisioning, task classification and allocation, resource release

and update, and inference learning are all performed by this module. The provisioning process starts with an estimation of the required initial VMs. The initial VMs' capacity for the vCell is computed from the implementation of the Capacity equation [61]. The capacity is then used as input to XEN API libraries to create the required VMs. The Provisioner classifies the submitted GSE data into groups using an implementation of dynamic K-Means strategy shown in the paper [61]. After the classification, the function $getSuitableHost()$ is called to invoke the host service wrapper class (HostServiceWrapper). The HostServiceWrapper replies with the set of available physical hosts and their available capacities. pVM then chooses the most suitable as guided by the SLA parameters in vCell's submitted request.

After the selection of a suitable host, the required Xen Api (XEN) VM class is invoked to create all the required initial VMs. The VIF class instance is then invoked to create a network of links and virtual interfaces. One virtual interface is created for each network definition in the VXDL request file. Using the initial task inference from the dynamic Markov chain model, an initial virtual interface is selected as the default for communication. The allocated bandwidth to a VM is constantly updated by the pVM to reflect the various phases of job execution at the VM. This process ensures that available bandwidths in the vCell are properly budgeted.

Each created VM is started via the implementation of a call to the VM.start method. The VMs are configured with a standard socket to listen, on a specific port, for any incoming job submitted by pVM. The class RScript starts the R script, if it is not running, and sets the current working directory of the R workspace to the directory where the data to be analysed is saved. At the end of computation the results are written back to the VM class which subsequently writes the final result to the MySQL database server.

We implement virtual file service (VFS) functionalities in combination with allocation and control to provide an effective and flexible storage server. Our approach attempts to minimize the IO overhead overtime caused by VM sprawl. Each VM is enabled with a

microarray data analysis algorithm and accesses the required files in a “just in time policy. This approach allows ease in relocation of jobs since only the required data is copied and the relocation to any idle resource is easily achieved.

The pVM periodically checks all running instances and decides, based on the status information (failed, running, idle, halt) obtained, whether to reassign the job. The functionalities provided include: initiate termination of a VM, report reclaim resources to resident vCell manager, and initiate creation of new virtual resource. This way, the job status can be determined and, where necessary, relocation of the job to a new instance initiated. The set-up determines job status by computing completion time as a function of GSE folder size, available memory in allocated VM, and data transfer delay. After the expected finished time, the VM is marked inaccessible and the job running in the VM marked as failed. If status is failed, the module notifies the provisioning module which then destroys the inaccessible VM, assigns the destroyed VMs resources to create a new VM, and finally reassigns the failed job to the newly created VM.

D.2 Algorithms for job allocation

The algorithms for job allocation considered in this work will now be presented. The use of container based resource provisioning enables the sharing of statistics between the vCell manager and other resources in the vCell. The virtual machines are allocated bandwidth to satisfy the current job requirement (computed from time series). At the end of the data access time, the bandwidth allocated is reconfigured to allow other VMs to utilize the container’s overall capacity.

D.2.1 SJF-KQ

The Shortest Job First algorithm on K Queues (SJF-KQ) is a variation of Shortest Job First (SJF). In this approach, jobs are ordered in decreasing order of size and submitted to the

suitable category, K, of VMs created during the provisioning stage. This algorithm executes the submitted jobs based on the resource queues implemented in the provisioning phase.

D.2.2 SJF-KQ-L

The Shortest Job First algorithm on K Queues with Lookup (SJF-KQ-L) is a variation of SJF-KQ. However, the expected finish time F of each job is utilized to vary the data access period of the jobs. If the data access time of the job *i* is T_i, then the input/output overhead at storage server contributed by *i* at each point in F_i - T_i is zero. This variation on data access is exploited to reduce the overheads associated with data access by the virtual machines during jobs execution. The algorithm carefully schedules the job execution of VMs to ensure that the number of concurrent VMs accessing the NFS server is reduced. This way, the makespan is consequently reduced.

D.2.3 FCFS-KQ-L

The First Come First Served algorithm on K Queues with Lookup (FCFS-KQ-L) is a variation of SJF-KQ-L that is based on the widely used First Come First Served (FCFS) discipline instead of the SJF.

D.3 Results

The work investigates the impact of adopting cohesive operational behaviours among the VMs to exploit the differences in data access times and implement effective resource provisioning and job scheduling. Each virtual machine requests bandwidth of a certain size to satisfy the job submitted. After the duration of data access (the reading of data files for analysis program use), the virtual machine's bandwidth is reduced to the basic bandwidth. This is achieved by inserting a new action table entry in the software-defined enabled virtual switch. The residual bandwidth from the reconfiguration is made available for other VMs.

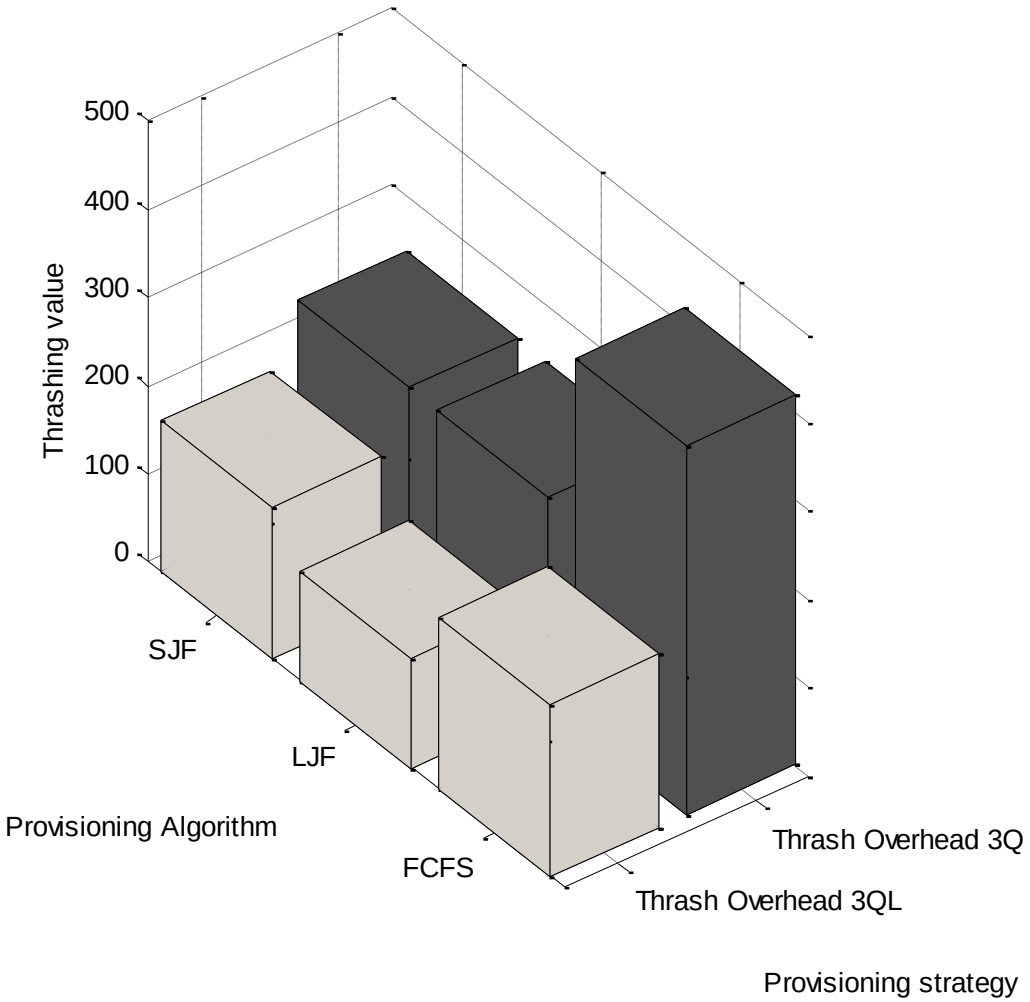


Figure D.2: Result comparing the thrashing rate between the proposed algorithms using common provisioning algorithms

The first experiment investigates the performance of the proposed provisioning algorithms. The algorithm that utilizes the predicted finished time lookup during provisioning is compared using three well-known [73] on-demand algorithms for virtual machines provisioning. Figure D.2 shows that the proposed algorithms reduced the thrashing rate which in turn reduces the frequency of creation and destruction of virtual machines. A high thrashing rate increases the workload on a provisioner and the instability of the data analysis process as resources are created and released. Since cloud services are normally charged per hour, the creation and release of virtual machines incur additional overhead costs. Hence, a small

value for the thrashing rate is required to maintain a stable and, consequently, cost effective job execution. We use First Come First Serve (FCFS), Largest Job First (LJF), and Shortest Job First (SJF) to demonstrate (Figure D.2) that predicting the finish time and taking the prediction into consideration during the classification of jobs into groups during resource provisioning phase reduces the thrashing rate.

In all the experiments, we set the value of K=3 for the algorithms presented above. For example example, LJF-KQL becomes LJF-3QL. The result in Figure D.2 shows that the algorithms ('Thrash overhead 3QL') implemented with a lookup outperformed those without lookup ('Thrash overhead 3Q'). In the experiment, a maximum of ten virtual machines and one privilege VM instances are instantiated per vCell. The budget size for the chosen experiment is set to $\$100$. The experiment involved the analysis of $30GB$ of microarray data. The values on the y-axis in figure D.2 show the number of times a VM is released and a new one created to accommodate a new analysis job.

In Figure D.2, all three algorithms that enhanced common provisioning disciplines (FCFS, SJF, and LJF) with a group classification and finished time lookup outperformed those without such enhancement. This is possible as each VM in the vCell operates alongside members of the vCell as a complementary component. Note that achieving such enhanced provisioning is made possible due to the small size of the vCell. Implementing the same finished time lookup and the grouping of jobs to complement each other at the whole datacentre level will amount to large computation savings.

We continue with a comparison between our modified versions of SJF task allocation algorithms as SJF-3Q and SJF-3Q-L respectively. Figure D.3 shows the result obtained from analyzing up to $30GB$ of microarray data. We first defined the makespan as the time difference between the start and finish of the data analysis and included time to read the required data, perform computation to identify gene expression, and write the output result to the DBS. In the figure, SJF-3Q-L outperformed the other two algorithms with shorter makespan at various job sizes. We attribute this higher performance to the ability of the

vCell manager to assign jobs based on expected finished times.

We then investigate the impact of using our proposed holistic view of cloud resources on the cost of data analysis. As demonstrated in Figure D.4, the cost of analysing the data is lower for algorithms utilizing the finish times of jobs on virtual machines. We attribute the performance strength of SJF-3Q-L to two features:

- the small size of VM thrashing reduces the cost of resource usage since VMs are only created gradually as resources are released by the large number of small VMs.
- by carefully utilizing the jobs' statistics from the report interface module, we can allocate jobs in a way that reduces concurrent data access and improves the performance.

In summary, the combined results in Figures D.2, D.3, and D.4 highlight the fact that the holistic view of resources improved the performance of well-known algorithms for resource provisioning and job allocation. Quantitatively, the cost of performing analysis is reduced by 30% at 15GB of data analysis. The makespan also reduces by more than half an hour at 15GB of data analysis.

D.4 Conclusion

There is a general perception [74, 75] that the next wave of cloud services are to be dominated by PAAS value added services. The virtual container described in this work is a step toward this advancement. In our case, the value added service is created by the bioinformatician and packaged as a virtual machine. The scientist performing the experiment requests this virtual machine and other required resources as a self-service and dynamic container. Our work demonstrates the strength of this next generation cloud framework in performing cost effective analysis of microarray data on commodity hardware. Flexibility, dynamic configuration, and elasticity were enabled by creating a self-service infrastructure container which allows the scientist performing the experiment to submit an abstract description of requirements. Furthermore, the cloud framework proposed in this work allows VMs to

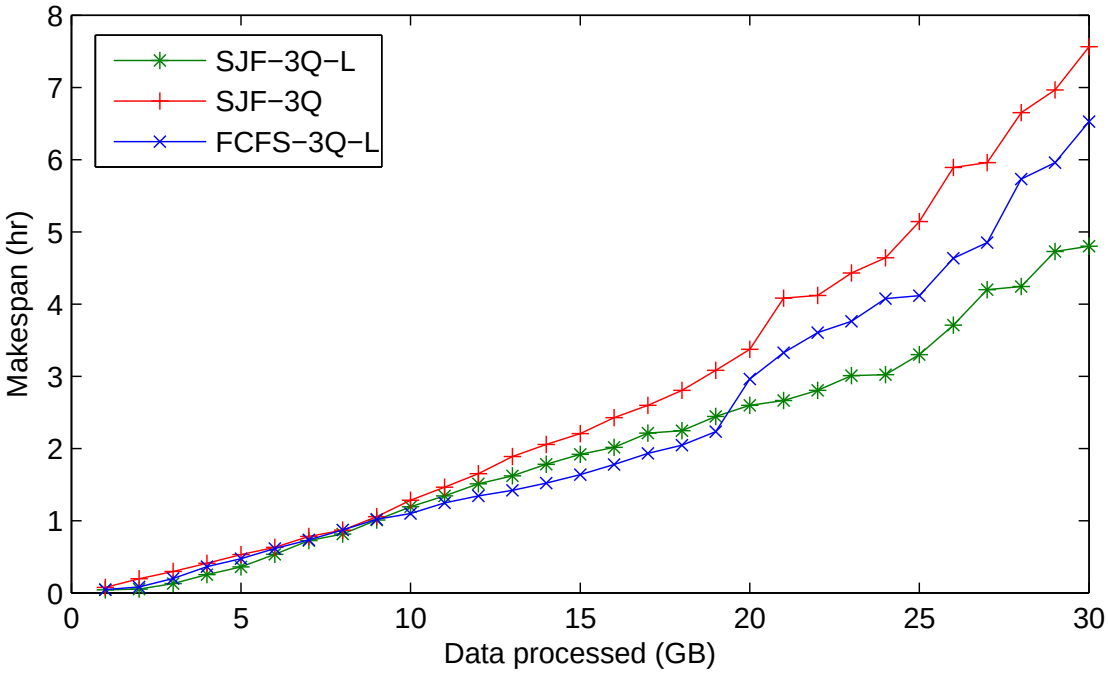


Figure D.3: Result comparing the job makespan between the proposed job allocation algorithm and other algorithms

operate in concert with each other and with the enabling logic. The dynamic feature of the model reduces the need to understand technical cloud computing concepts.

The virtual container presented in this work is enabled with Markov Chain learning and prediction that allows the container to manage itself using previous observations from job execution traces. We use the automation capability to estimate initial VMs' capacity without the intervention of a user.

This article demonstrates the concept in a prototype experimental cloud built on commodity hardware. The cloud environment is created using XEN Cloud Platform (XCP). The proposed privileged virtual machine is equipped with necessary XAPI compliant java modules.

A significant difference between the strategy described in this work and existing clouds is the holistic view of the resource. Also in the proposed framework, use of an observed pattern of data analysis is applied to automate the whole data analysis process. Using the variation in instant virtual machine bandwidth requirements, our proposed algorithms

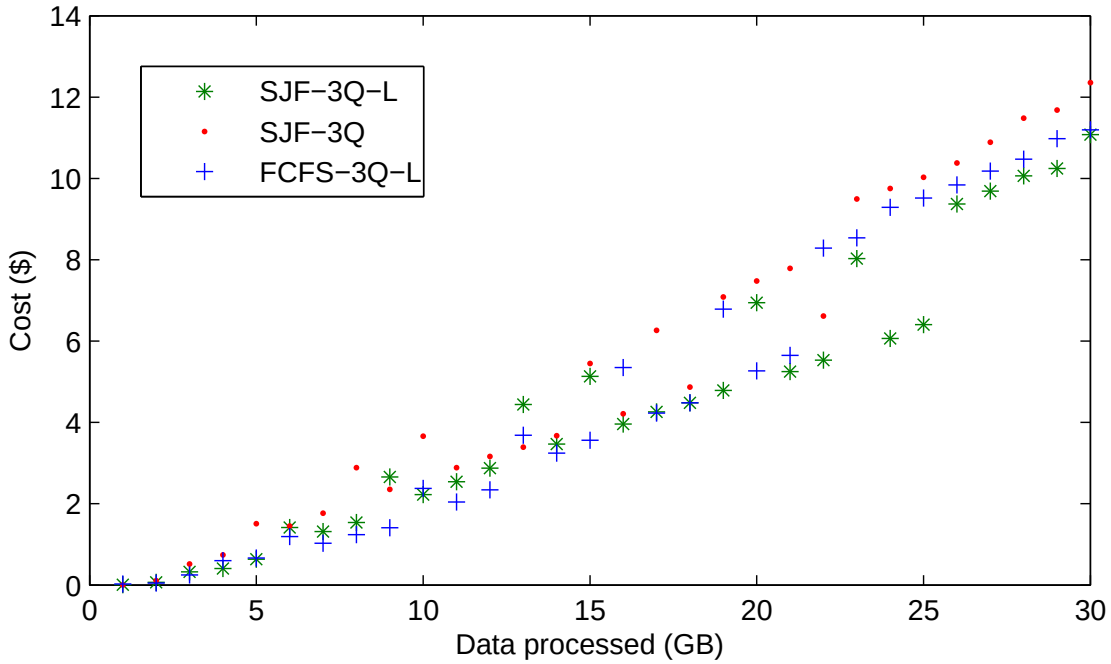


Figure D.4: Cost comparison among various job allocation algorithms

improved the performance and led to considerable reduction in cost at a performance that guarantees the same experience as commercial cloud services. Although this work focused on a data intensive cloud application, the same logic can easily be extended to other cloud applications. In the future, our work aims to implement the same container model for parallel and distributed cloud applications.

Appendix E

Venus-C Project

Venus-C is a project funded under the European Commission's 7th Framework Programme. It consists of a joint co-operation between computing service providers such as Microsoft and scientific user communities to develop, test and deploy a large Cloud Computing infrastructure for science researchers and SMEs (Small and Medium size Enterprises) in Europe. Venus-C began with seven Partner User Scenarios, or partner affiliations, from academic, research or small commercial organisations. They developed applications on the Windows Azure Cloud in the areas of bioinformatics, systems biology, drug discovery, civil engineering, civil protection and emergencies, and marine biodiversity data. An early view of the Venus-C website can be seen in Figure E.1.

A collaboration between Dr Andrew Harrison of the University of Essex and Dr Hugh Shanahan of Royal Holloway College resulted in the acceptance of their Open Call proposal to be one of the 2nd tranch of projects, called Venus-C pilot projects. Their pilot project, called cTQm (**c**loud **T**ool for **Q**uality control of **m**icroarray data), was to develop a cloud based tool for testing the quality of microarray data, initially from *H. Sapiens*. The work done for this project forms the basis for this chapter and for the following chapter on Wide scale analysis.

During the one year project of the Venus-C pilots, processing and storage facilities on the Windows Azure Cloud was free to the pilot projects, together with technical support

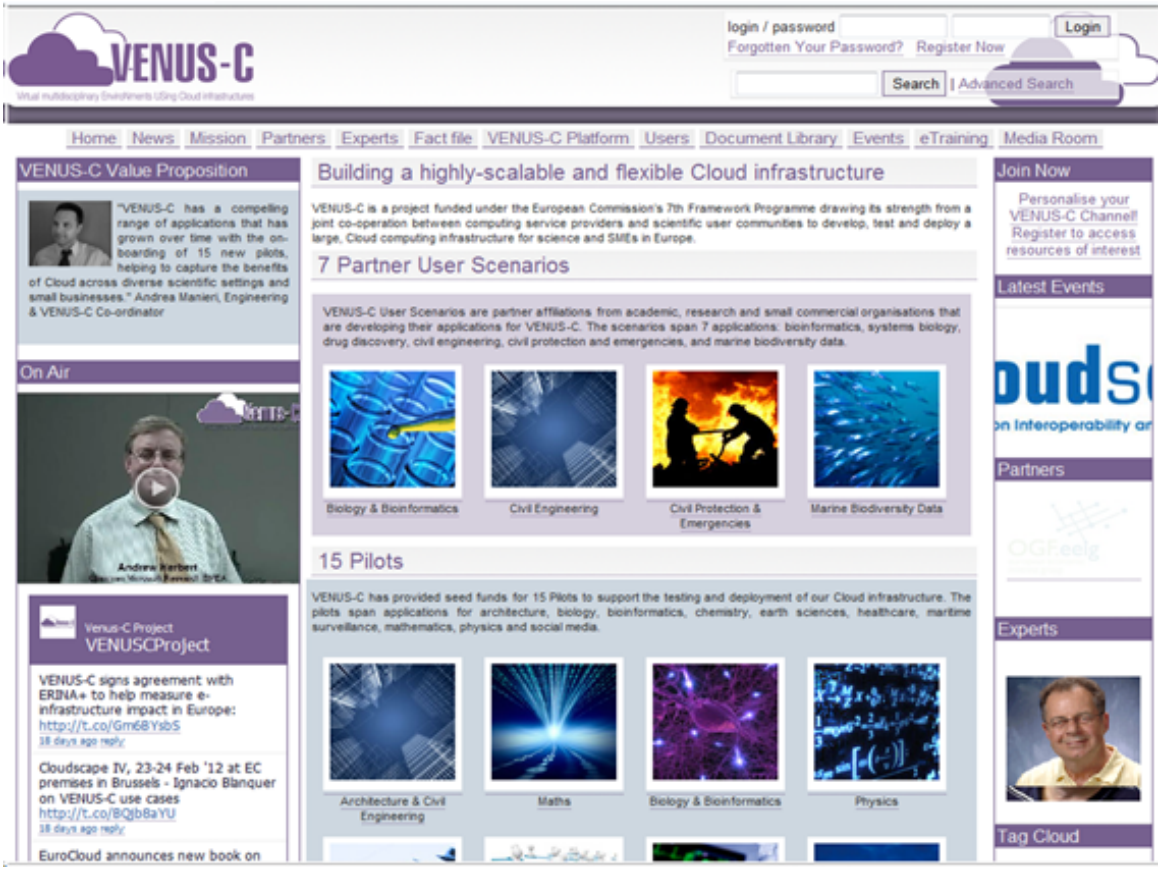


Figure E.1: The Venus-C website home page from June, 2012, showing where Biology and Bioinformatics were the subject areas of some of the 15 pilot projects as well as some of the original 7 partner scenarios

from Microsoft and the Venus-C team. During the following year, which ended on May 31st, 2013, the use of certain facilities of the Windows Azure Cloud were available without charge to the pilot projects, but without technical support.

It was envisaged that the following tasks could usefully be carried out to further the research into improving the quality of microarray data:

1. Scripts already developed in R, using related Bioconductor libraries, for analyzing the effect of G-stacks on normalized microarray data, would be ported to the Azure cloud.

2. The Azure cloud would be used to repeat the analysis of G-stack bias on normalized data in individual experiments over some HG_U133A datasets, performed by Shana-

han *et al* [38]. Then the analysis would be extended to all available experimental data from this human GeneChip. Then several other microarray designs, e.g. another human genechip HG_U133_Plus2 and *Arabidopsis thaliana*, would be investigated.

3. The Azure cloud would be used to examine the size of bias that G-stacks have on correlations between transcripts over multiple experiments (which is a key variable in Systems Biology studies).

4. Some easy to use front end software could then be developed for wet lab Biologists to make use of these tools.

Appendix F

Poster

There follows the poster called **G-stack bias in publicly deposited Affymetrix HG_U133A Microarray data** which was presented at the 21st Annual International Conference on Intelligent Systems for Molecular Biology (ISMB) in July 2013, in Berlin, Germany. This conference was held in conjunction with the 12th European Conference on Computational Biology (ECCB).

G-stack bias in publicly deposited Affymetrix HG-U133A Microarray data

Anne M. Owen, Department of Mathematical Sciences, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK
Hugh Shanahan, Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK
Andrew P. Harrison, Department of Mathematical Sciences, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK

Contact : Hugh.Shanahan@rhul.ac.uk
W : http://www.shanahanlab.org/~hugh
T : +44 (0)1784 443433

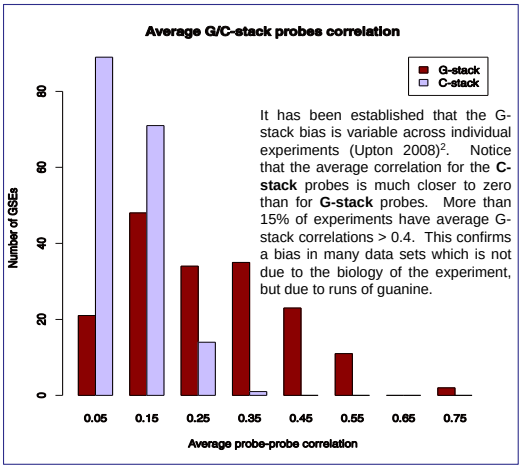
G-stack in a probe sequence

Introduction

It has been shown (Shanahan 2011)[1] that normalized Affymetrix expression data are biased by G-quadruplex formation. In that paper a limited subset of all the publicly deposited HG-U133A GeneChip microarray data was investigated. This work has undertaken a wide scale survey of all Affymetrix HG-U133A microarray data available on ArrayExpress at July 2012. Two types of analysis have been used to show the overall extent of the bias.

The computing analyses have been carried out in R on the Windows Azure Cloud. This demonstrates the vitally important part that cloud computing can play in bioinformatics research. The bringing together of large datasets and scalable computing power enables wide scale surveys such as this one to be done.

Microarrays are concerned with probes which are organised into probe sets to reveal the expression of particular genes. A probe which has four or more contiguous 'G' (guanine) bases in its sequence is defined as a **G-stack**. Similarly a **C-stack** probe has a sequence containing four or more contiguous 'C' (cytosine) bases.



It has been established that the G-stack bias is variable across individual experiments (Upton 2008)[2]. Notice that the average correlation for the **C-stack** probes is much closer to zero than for **G-stack** probes. More than 15% of experiments have average G-stack correlations > 0.4. This confirms a bias in many data sets which is not due to the biology of the experiment, but due to runs of guanine.

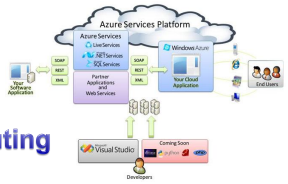
Big Data

The Windows Azure Cloud was used, because the large amount of data involved could be stored in Azure Blob Storage, close to the processing facility. The first task was to upload hundreds of Affymetrix GeneChip experiment results from the publicly available ArrayExpress database to Azure storage. This was done via a webpage serviced by a WebRole task in Azure. R was used for the main script with service calls in C# to create folders and files as required. Problems of inconsistency in the data were overcome individually.

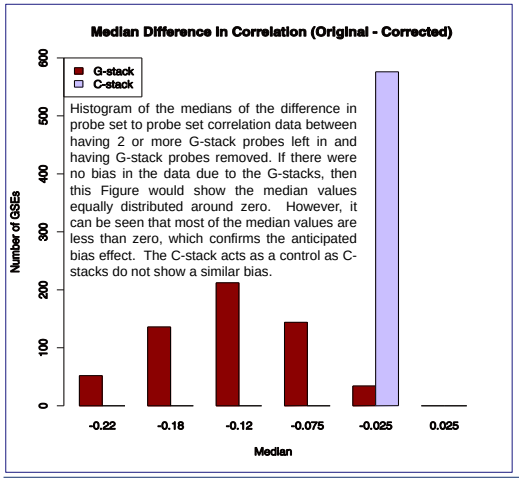
- **Large survey of Affymetrix HG-U133A data**
- **Azure Cloud Computing used with R**
- **Evidence of widespread G-stack bias proven**
- **As many as 15% experiments affected by bias**
- **Software for bias detection in development**

Methods

Nearly 600 Affymetrix GeneChip datasets from the HG-U133A array were used. Programming was carried out with R scripts and Bioconductor modules such as *affy*. Chip Definition Files (CDFs) were modified to eliminate certain probes or probe sets as required to test the effect of eliminating G-stacks or C-stacks.



Cloud Computing



Histogram of the medians of the difference in probe set to probe set correlation data between having 2 or more G-stack probes left in and having G-stack probes removed. If there were no bias in the data due to the G-stacks, then this Figure would show the median values equally distributed around zero. However, it can be seen that most of the median values are less than zero, which confirms the anticipated bias effect. The C-stack acts as a control as C-stacks do not show a similar bias.

Conclusions

- The G-stack bias in Affymetrix microarray data is widespread in experiments deposited in public databases. It has been shown to be significant in at least 15% of HG-U133A experiments available in public databases in July 2012.
- The Azure Cloud is a useful tool for Bioinformaticians who prefer Windows to Linux, considering its scalability for repeating statistical analyses over large amounts of data. The use of R scripts and Bioconductor modules has been demonstrated on Azure's Platform as a Service (PaaS) cloud.

References

[1] Shanahan H.P., Memon F.N., Upton G.J.G. and Harrison A.P. (2011) 'Normalized Affymetrix expression data are biased by G-quadruplex formation' *Nucleic Acids Research, 2011,* ***1-9***

[2] Upton G.J.G., Langdon W., and Harrison A.P. (2008) 'G-spots cause incorrect expression measurement in Affymetrix microarrays' *BMC Genomics, 2008, 9, 613*

Acknowledgements

European Commission's 7th Framework Programme
Microsoft and Venus-C project Organisers

Department of Computer Science
Royal Holloway, University of London
Egham
TW20 0EX
England, U.K.
T: +44 (0) 1784 443431
F: +44 (0) 1784 439786
http://www.cs.rhul.ac.uk

Department of Mathematical Sciences
University of Essex, Wivenhoe Park
Colchester, Essex
CO4 3SQ
England, U.K.
T: +44 (0) 1206 873040
F: +44 (0) 1206 873043
http://www.essex.ac.uk

Venus-C
Virtual multidisciplinary EnviroNments USing Cloud infrastructures

Bibliography

[1] Rafael A Irizarry et al. “Exploration, normalization, and summaries of high density oligonucleotide array probe level data.” In: *Biostatistics (Oxford, England)* 4.2 (Apr. 2003), pp. 249–64. ISSN: 1465-4644. DOI: 10.1093/biostatistics/4.2.249. URL: http://biostatistics.oxfordjournals.org/cgi/content/long/4/2/249.

[2] Affymetrix. “Guide to PLIER Estimation”. In: *Affymetrix Technical Note* (Aug. 2005). URL: http://media.affymetrix.com/support/technical/technotes/plier_technote.pdf.

[3] W B Langdon et al. “A Survey of Spatial Defects in Homo Sapiens Affymetrix GeneChips”. English. In: *Ieee-Acm Transactions on Computational Biology and Bioinformatics* 7.4 (2010), pp. 647–653. DOI: Doi10.1109/Tcbb.2008.108. URL: <GotoISI>://WOS:000283559100009.

[4] G J G Upton and A P Harrison. “Motif effects in Affymetrix GeneChips seriously affect probe intensities”. English. In: *Nucleic Acids Research* 40.19 (2012), pp. 9705–9716. DOI: Doi10.1093/Nar/Gks717. URL: <GotoISI>://WOS:000310377200036.

[5] Mark J. Costello, Robert M. May, and Nigel E. Stork. “Can We Name Earth’s Species Before They Go Extinct?” In: *Science* 339.6118 (2013), pp. 413–416. DOI: 10.1126/science.1230318. eprint: http://www.sciencemag.org/

content/339/6118/413.full.pdf. URL: http://www.sciencemag.org/content/339/6118/413.abstract.

[6] Chris J Needham et al. "From gene expression to gene regulatory networks in Arabidopsis thaliana." In: *BMC Systems Biology* 3.1 (Sept. 2009), p. 85. ISSN: 1752-0509. DOI: 10.1186/1752-0509-3-85. URL: http://www.biomedcentral.com/1752-0509/3/85.

[7] Belinda Giardine et al. "Galaxy: A platform for interactive large-scale genome analysis". In: *Genome Research* 15 (2005), pp. 1451–1455. ISSN: 10889051. DOI: 10.1101/gr.4086505.

[8] Dunca Hull et al. "Taverna: A tool for building and running workflows of services". In: *Nucleic Acids Research* 34 (2006), pp. 729–732. ISSN: 03051048. DOI: 10.1093/nar/gkl320.

[9] Mohamed Abouelhoda, Shadi A Issa, and Moustafa Ghanem. "Tavaxy: Integrating Taverna and Galaxy workflows with cloud computing support." In: *BMC bioinformatics* 13.1 (May 2012), p. 77. ISSN: 1471-2105. DOI: 10.1186/1471-2105-13-77. URL: http://www.biomedcentral.com/1471-2105/13/77.

[10] Nicholas M Luscombe, Dov Greenbaum, and Mark Gerstein. "What is bioinformatics? An introduction and overview". In: *Yearbook of Medical Informatics* 1 (2001), pp. 83–99. URL: http://www.ebi.ac.uk/luscombe/docs/imia_review.pdf.

[11] Kimberley Robasky, Nathan E. Lewis, and George M. Church. "The role of replicates for error mitigation in next-generation sequencing". In: *Nature Reviews Genetics* 15.15 (Dec. 2013), pp. 56 –62. DOI: 10.1038/nrg3655. URL: http://www.nature.com/nrg/journal/v15/n1/full/nrg3655.html#concluding-remarks.

[12] G J G Upton, W B Langdon, and A P Harrison. "G-spots cause incorrect expression measurement in Affymetrix microarrays". English. In: *Bmc Genomics* 9 (2008). DOI: Artn613Doi10.1186/1471-2164-9-613. URL: <GotoISI>://WOS:000263109600001.

[13] Vivien Marx. "Biology: The big challenges of big data." In: *Nature* 498.7453 (June 2013), pp. 255–60. ISSN: 1476-4687. DOI: 10.1038/498255a. URL: http://dx.doi.org/10.1038/498255a.

[14] EMBL-EBI. *EMBL Annual Report 2012-2013*. 2013.

[15] Manolis Kellis et al. "Defining functional DNA elements in the human genome". In: *Proceedings of the National Academy of Sciences* 111.17 (2014), pp. 6131–6138. DOI: 10.1073/pnas.1318948111. eprint: http://www.pnas.org/content/111/17/6131.full.pdf+html. URL: http://www.pnas.org/content/111/17/6131.abstract.

[16] Siegfried Schloissnig, Manimozhiyan Arumugam, and Shinichi et. al. Sunagawa. "Genomic variation landscape of the human gut microbiome". In: *Nature* 493.7430 (Jan. 2013), pp. 45–50. DOI: doi:10.1038/nature11711. URL: http://www.nature.com/nature/journal/v493/n7430/full/nature11711.html.

[17] Nick Goldman, Paul Bertone, and Siyuan et. al. Chen. "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA". In: *Nature* 494.1038 (Feb. 2013), pp. 77–80. DOI: doi:10.1038/nature11875. URL: http://www.nature.com/nature/journal/v494/n7435/full/nature11875.html.

[18] Michael Specter. "Letter from Shenzhen, "The Gene Factory"". In: *The New Yorker* 2014.Jan 6 (2014), pp. 34–34. URL: http://www.newyorker.com/reporting/2014/01/06/140106fa_fact_specter.

[19] I. Foster et al. “Cloud Computing and Grid Computing 360-Degree Compared”. In: *Grid Computing Environments Workshop, 2008. GCE ’08*. Nov. 2008, pp. 1–10. DOI: 10.1109/GCE.2008.4738445.

[20] D. Kondo et al. “Cost-benefit analysis of Cloud Computing versus desktop grids”. In: *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. May 2009, pp. 1–12. DOI: 10.1109/IPDPS.2009.5160911.

[21] Ian Foster and Carl Kesselman. “What is the Grid”. In: *A three point checklist* 20 (2002).

[22] Klaus Krauter, Rajkumar Buyya, and Muthucumaru Maheswaran. “A taxonomy and survey of grid resource management systems for distributed computing”. In: *Software: Practice and Experience* 32.2 (2002), pp. 135–164. ISSN: 1097-024X. DOI: 10.1002/spe.432. URL: http://dx.doi.org/10.1002/spe.432.

[23] Wolfgang Hoschek et al. “Data Management in an International Data Grid Project”. English. In: *Grid Computing GRID 2000*. Ed. by Rajkumar Buyya and Mark Baker. Vol. 1971. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000, pp. 77–90. ISBN: 978-3-540-41403-2. DOI: 10.1007/3-540-44444-0_8. URL: http://dx.doi.org/10.1007/3-540-44444-0_8.

[24] Ann Chervenak et al. “The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets”. In: *Journal of Network and Computer Applications* 23.3 (2000), pp. 187 –200. ISSN: 1084-8045. DOI: http://dx.doi.org/10.1006/jnca.2000.0110. URL: http://www.sciencedirect.com/science/article/pii/S1084804500901103.

[25] Peter Mell and Timothy Grance. “The NIST Definition of Cloud Computing”. In: *NIST Special Publication* 800-145 (Sept. 2011), p. 2. URL: csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[26] Dan C. Marinescu. *Cloud Computing: Theory and Practice*. 225 Wyman St. Waltham 02451 USA: Elsevier Inc., 2013. ISBN: 978-0-12404-627-6.

[27] Rita JM Volkers et al. "Gene-environment and protein degradation signatures characterize genomic and phenotypic diversity in wild Caenorhabditis elegans populations." In: *BMC biology* 11.1 (Aug. 2013), p. 93. ISSN: 1741-7007. DOI: 10.1186/1741-7007-11-93. URL: http://www.biomedcentral.com/1741-7007/11/93.

[28] Keiichi Kodama et al. "Expression-based genome-wide association study links the receptor CD44 in adipose tissue with type 2 diabetes." In: *Proceedings of the National Academy of Sciences of the United States of America* 109.18 (May 2012), pp. 7049–54. ISSN: 1091-6490. DOI: 10.1073/pnas.1114513109. URL: http://www.pnas.org/content/109/18/7049.short.

[29] Laurent Gautier et al. "affy - analysis of Affymetrix GeneChip data at the probe level". In: *Bioinformatics* 20.3 (2004), pp. 307–315. DOI: 10.1093/bioinformatics/btg405. URL: http://www.researchgate.net/publication/62081-06_affy--analysis_of_Affymetrix_GeneChip_data_at_the_probe_level/file/5046351543b4294184.pdf&sa=X&scisig=AAGBfm31-XxSg1tARh6kZ5zpA-fJx4c47A&oi=scholarr&ei=o1mEU-7LJJsfI0w-XdgoCACg&ved=0CEEQgAMoAjAA.

[30] T Bammler and Beyer RP et. al. "Standardizing global gene expression analysis between laboratories and across platforms". In: *Nature Methods* 2.5 (May 2005), pp. 351–356. URL: http://www.ncbi.nlm.nih.gov/pubmed/15846362.

[31] J. Augen. *Bioinformatics in the post-genomic era: genome, transcriptome, proteome, and information-based medicine*. Addison-Wesley, 2005. ISBN: 9780321173867. URL: http://books.google.co.uk/books?id=IhBFAQAAIAAJ.

[32] Graham J G Upton et al. “On the causes of outliers in Affymetrix GeneChip data.” In: *Briefings in functional genomics & proteomics* 8.3 (May 2009), pp. 199–212. ISSN: 1477-4062. DOI: 10.1093/bfgp/elp027. URL: http://www.ncbi.nlm.nih.gov/pubmed/19734302.

[33] Zhijin WU et al. “A model-based background adjustment for oligonucleotide expression arrays”. eng. In: *Journal of the American Statistical Association* 99.468 (2004), pp. 909–917. ISSN: 0162-1459. URL: http://cat.inist.fr/?aModele=afficheN\&cpsidt=16304649.

[34] C. Li and W. H. Wong. “Model-based analysis of oligonucleotide arrays: Expression index computation and outlier detection”. In: *Proceedings of the National Academy of Sciences* 98.1 (Jan. 2001), pp. 31–36. ISSN: 0027-8424. DOI: 10.1073/pnas.98.1.31. URL: http://www.pnas.org/cgi/content/long/98/1/31.

[35] Tanya Barrett and Ron Edgar. “Mining microarray data at NCBI’s Gene Expression Omnibus (GEO)*.” In: *Methods in molecular biology (Clifton, N.J.)* 338 (Jan. 2006), pp. 175–90. ISSN: 1064-3745. DOI: 10.1385/1-59745-097-9:175. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1619899\&tool=pmcentrez\&rendertype=abstract.

[36] A. Brazma. “ArrayExpress–a public repository for microarray gene expression data at the EBI”. In: *Nucleic Acids Research* 31.1 (Jan. 2003), pp. 68–71. ISSN: 13624962. DOI: 10.1093/nar/gkg091. URL: http://nar.oxfordjournals.org/cgi/content/long/31/1/68.

[37] Maria A Stalteri and Andrew P Harrison. “Interpretation of multiple probe sets mapping to the same gene in Affymetrix GeneChips.” In: *BMC bioinformatics* 8.1 (Jan. 2007), p. 13. ISSN: 1471-2105. DOI: 10.1186/1471-2105-8-13. URL: http://www.biomedcentral.com/1471-2105/8/13.

[38] Hugh P Shanahan et al. "Normalized Affymetrix expression data are biased by G-quadruplex formation." In: *Nucleic Acids Research* 40.8 (Apr. 2012), pp. 3307–3315. ISSN: 1362-4962. DOI: 10.1093/nar/gkr1230. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3333884\&tool=pmcentrez\&rendertype=abstract.

[39] Rafael A Irizarry, Zhijin Wu, and Harris A Jaffee. "Comparison of Affymetrix GeneChip expression measures." In: *Bioinformatics (Oxford, England)* 22.7 (Apr. 2006), pp. 789–94. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btk046. URL: http://bioinformatics.oxfordjournals.org/cgi/content/long/22/7/789.

[40] Federico M Giorgi et al. "Algorithm-driven artifacts in median Polish summarization of microarray data." In: *BMC bioinformatics* 11.1 (Jan. 2010), p. 553. ISSN: 1471-2105. DOI: 10.1186/1471-2105-11-553. URL: http://www.biomedcentral.com/1471-2105/11/553.

[41] Sepp Hochreiter, Djork-Arné Clevert, and Klaus Obermayer. "A new summarization method for Affymetrix probe level data." In: *Bioinformatics (Oxford, England)* 22.8 (Apr. 2006), pp. 943–9. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btl033. URL: http://bioinformatics.oxfordjournals.org/cgi/content/long/22/8/943.

[42] Anon. "Affymetrix Platform compared to Stanford Platform". In: *Expression Analysis* (2005), pp. 1–10. URL: www.osa.sunysb.edu/udmf/Affy-Platform-Comparison-Tech-Note.pdf.

[43] B M Bolstad et al. "A comparison of normalization methods for high density oligonucleotide array data based on variance and bias". In: *Bioinformatics* 19.2 (Jan. 2003), pp. 185–193. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/19.2.

185. URL: http://bioinformatics.oxfordjournals.org/cgi/content/abstract/19/2/185.

[44] Affymetrix. "Statistical Algorithms Description Document". In: *Affymetrix Document* (2002), p. 28. URL: https://madb.nci.nih.gov/mAdb-public/Training/affymetrix_sadd_whitepaper.pdf.

[45] W. J. Lemon et al. "Theoretical and experimental comparisons of gene expression indexes for oligonucleotide arrays". In: *Bioinformatics* 18.11 (Nov. 2002), pp. 1470–1476. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/18.11.1470. URL: http://bioinformatics.oxfordjournals.org/cgi/content/long/18/11/1470.

[46] Rehannah Borup et al. "Development and production of an oligonucleotide MuscleChip: use for validation of ambiguous ESTs". In: *BMC Bioinformatics* 3.1 (2002), p. 33. ISSN: 1471-2105. DOI: 10.1186/1471-2105-3-33. URL: http://www.biomedcentral.com/1471-2105/3/33.

[47] Ravi Kothapalli et al. "Microarray results: how accurate are they?" In: *BMC Bioinformatics* 3.1 (2002), p. 22. ISSN: 1471-2105. DOI: 10.1186/1471-2105-3-22. URL: http://www.biomedcentral.com/1471-2105/3/22.

[48] F. Mosteller eds D.C. Hoaglin and J. W. Tukey. *Understanding Robust and Exploatory Data Analysis*. New York: John Wiley & Sons, 1983, pp. 165 –210. ISBN: 0-471-38491-7.

[49] A. Badiee et al. "Evaluation of five different cDNA labeling methods for microarrays using spike controls". In: *BMC Biotechnology* 3.23 (Dec. 2003). DOI: 10.1186/1472-6750-3-23. URL: http://www.biomedcentral.com/1472-6750/3/23.

[50] Terry M. Therneau and Karla V. Ballman. "What does PLIER really do?" In: *Cancer Informatics* 6.6 (Aug. 2008), pp. 423–431. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2623311/.

[51] Olivia Sanchez-Graillet et al. "Widespread existence of uncorrelated probe intensities from within the same probeset on Affymetrix GeneChips". In: *Journal of Integrative Bioinformatics* 5.2 (2008), p. 13. URL: http://journal.imbio.de/article.php?aid=98.

[52] S F Altschul et al. "Basic local alignment search tool." In: *Journal of molecular biology* 215.3 (Oct. 1990), pp. 403–10. ISSN: 0022-2836. DOI: 10.1016/S0022-2836(05)80360-2. URL: http://www.sciencedirect.com/science/article/pii/S0022283605803602.

[53] Chunlei Wu et al. "Short oligonucleotide probes containing G-stacks display abnormal binding affinity on Affymetrix microarrays." In: *Bioinformatics (Oxford, England)* 23.19 (Oct. 2007), pp. 2566–72. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btm271. URL: http://bioinformatics.oxfordjournals.org/cgi/content/long/23/19/2566.

[54] Haiyong Han and Laurence H Hurley. "G-quadruplex DNA: a potential target for anti-cancer drug design". In: *Trends in Pharmacological Sciences* 21.4 (Apr. 2000), pp. 136–142. ISSN: 01656147. DOI: 10.1016/S0165-6147(00)01457-7. URL: http://www.sciencedirect.com/science/article/pii/S0165614700014577.

[55] Sarah Burge et al. "Quadruplex DNA: sequence, topology and structure." In: *Nucleic acids research* 34.19 (Jan. 2006), pp. 5402–15. ISSN: 1362-4962. DOI: 10.1093/nar/gkl655. URL: http://nar.oxfordjournals.org/cgi/content/long/34/19/5402.

[56] W B Langdon, G J G Upton, and A P Harrison. "Probes containing runs of guanines provide insights into the biophysics and bioinformatics of Affymetrix GeneChips". English. In: *Briefings in Bioinformatics* 10.3 (2009), pp. 259–277. DOI: Doi10.1093/Bib/Bbp018. URL: http://bib.oxfordjournals.org/content/early/2009/04/08/bib.bbp018.abstract.

[57] Farhat N Memon et al. "Identifying the impact of G-quadruplexes on Affymetrix 3' arrays using cloud computing." In: *Journal of integrative bioinformatics* 7.2 (Jan. 2010), p. 111. ISSN: 1613-4516. DOI: 10.2390/biecoll-jib-2010-111. URL: http://www.ncbi.nlm.nih.gov/pubmed/20134078.

[58] Ian Foster. "Globus Toolkit Version 4: Software for Service-Oriented Systems". English. In: *Journal of Computer Science and Technology* 21.4 (2006), pp. 513–520. ISSN: 1000-9000. DOI: 10.1007/s11390-006-0513-y. URL: http://dx.doi.org/10.1007/s11390-006-0513-y.

[59] Douglas Thain, Todd Tannenbaum, and Miron Livny. "Distributed computing in practice: the Condor experience." In: *Concurrency - Practice and Experience* 17.2-4 (2005), pp. 323–356.

[60] James C. Phillips et al. "Scalable molecular dynamics with NAMD". In: *Journal of Computational Chemistry* 26.16 (2005), pp. 1781–1802. ISSN: 1096-987X. DOI: 10.1002/jcc.20289. URL: http://dx.doi.org/10.1002/jcc.20289.

[61] Ibrahim K Musa et al. "Self-service infrastructure container for data intensive application". en. In: *Journal of Cloud Computing: Advances, Systems and Applications* 3.1 (May 2014), p. 5. ISSN: 2192-113X. DOI: 10.1186/2192-113X-3-5. URL: http://www.journalofcloudcomputing.com/content/3/1/5.

[62] T Hubbard et al. "The Ensembl genome database project". In: *Nucleic acids research* 30.1 (2002), pp. 38–41.

[63] Farhat N Memon et al. "Identifying the impact of G-quadruplexes on Affymetrix Exon arrays using Cloud Computing." In: *CIB2009* (2009). URL: http://www.cls.zju.edu.cn/binfo/IB/2009/Proceeding.pdf#page=55.

[64] Hugh P Shanahan, Anne M Owen, and Andrew P Harrison. "Bioinformatics on the cloud computing platform Azure." In: *PloS one* 9.7 (Jan. 2014). Ed. by Shyamal D. Peddada, e102642. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0102642. URL: http://dx.plos.org/10.1371/journal.pone.0102642.

[65] Sriram Krishnan. *Programming Windows Azure*. Sebastopol, CA: O'Reilly Media, Inc., 2010. ISBN: 978-0-596-80197-7.

[66] Sergio Contrino et al. "modMine: flexible access to modENCODE data." In: *Nucleic acids research* 40.Database issue (Jan. 2012), pp. D1082–8. ISSN: 1362-4962. DOI: 10.1093/nar/gkr921. URL: http://nar.oxfordjournals.org/cgi/content/long/40/D1/D1082.

[67] Yi Qu, Fei He, and Yuchen Chen. "Different effects of the probe summarization algorithms PLIER and RMA on high-level analysis of Affymetrix exon arrays." In: *BMC bioinformatics* 11 (Jan. 2010), p. 211. ISSN: 1471-2105. DOI: 10.1186/1471-2105-11-211. URL: http://www.biomedcentral.com/1471-2105/11/211/.

[68] Dimos Gaidatzis et al. "Overestimation of alternative splicing caused by variable probe characteristics in exon arrays". In: *Nucleic Acids Research* 37.16 (2009), e107. DOI: 10.1093/nar/gkp508. eprint: http://nar.oxfordjournals.org/content/37/16/e107.full.pdf+html. URL: http://nar.oxfordjournals.org/content/37/16/e107.abstract.

[69] Farhat Naureen Memon, Graham J G Upton, and Andrew P Harrison. "A Comparative Study of the Impact of G-Stack Probes on Various Affymetrix GeneChips

of Mammalia." In: *Journal of nucleic acids* 2010 (Jan. 2010). ISSN: 2090-021X. DOI: 10.4061/2010/489736. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2915844\&tool=pmcentrez\&rendertype=abstract.

[70] Farhat N Memon. "Study of the effects caused by the G-Quadruplex Structures on high-throughput nucleic acid measurements". In: *PhD Thesis at University of Essex* (2012), pp. 113, 149.

[71] Hanni Willenbrock et al. "Quantitative miRNA expression analysis : Comparing microarrays with next-generation sequencing". In: (2009), pp. 2028–2034. DOI: 10.1261/rna.1699809.454. URL: http://rnajournal.cshlp.org/content/15/11/2028.full.pdf+html.

[72] Affymetrix Developer Network. "Affymetrix CEL Data File Format". In: *2009 Affymetrix, Inc.* (2009). URL: http://media.affymetrix.com/support/developer/powertools/changelog/gcos-agcc/cel.html.

[73] Stéphane Genaud and Julien Gossa. "Cost-wait trade-offs in client-side resource provisioning with elastic clouds". In: *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE. 2011, pp. 1–8.

[74] Sergio Garcia-Gomez et al. "Challenges for the comprehensive management of Cloud Services in a PaaS framework". In: *Scalable Computing: Practice and Experience* 13.3 (2012).

[75] Yefin V Natis et al. "PaaS Road Map: A Continent Emerging". In: *Gartner Research* (2011).