Methods in Molecular Biology 1986

Springer Protocols

Verónica Bolón-Canedo
Amparo Alonso-Betanzos
*Editors*

# Microarray Bioinformatics

Humana Press

# METHODS IN MOLECULAR BIOLOGY

*Series Editor*
**John M. Walker**
**School of Life and Medical Sciences,**
**University of Hertfordshire, Hatfield,**
**Hertfordshire, AL10 9AB, UK**

For further volumes:
http://www.springer.com/series/7651

# Microarray Bioinformatics

Edited by

**Verónica Bolón-Canedo and Amparo Alonso-Betanzos**

*CITIC, Universidade da Coruña, A Coruña, Spain*

Humana Press

*Editors*
Verónica Bolón-Canedo
CITIC
Universidade da Coruña
A Coruña, Spain

Amparo Alonso-Betanzos
CITIC
Universidade da Coruña
A Coruña, Spain

ISSN 1064-3745 ISSN 1940-6029 (electronic)
Methods in Molecular Biology
ISBN 978-1-4939-9441-0 ISBN 978-1-4939-9442-7 (eBook)
https://doi.org/10.1007/978-1-4939-9442-7

# Preface

Over the last few decades, advances in molecular genetics technologies, such as DNA microarrays, have stimulated a new line of research in bioinformatics. DNA microarrays allow us to obtain a global view of the cell, where it is possible to measure the simultaneous expression of tens of thousands of genes. In particular, this type of data works by collecting information from tissue and cell samples regarding gene expression differences that could be useful for diagnosing disease or for distinguishing a specific tumor type.

Microarray data quickly became very popular among bioinformatics researchers. In a microarray experiment, there are usually very few samples (often fewer than 100 samples), but the number of features in the raw data ranges up to 60,000. This high dimensionality, together with the almost naturally unbalancedness of such data, makes the analysis of microarray data very appealing for machine learning and statistical researchers too.

This book provides a comprehensive review of the main, up-to-date methods, tools, and techniques for microarray data analysis. Internationally recognized experts address specific research topics and challenges in their areas of expertise, some of them being from the field of biology, others from the field of computer science, and others from the field of statistics. This interdisciplinarity provides valuable knowledge about the state-of-the-art methods for microarray analysis, covering the necessary steps for the acquisition of the data, its preprocessing, and its posterior analysis. From the field of biology, this book covers an introduction to bioinformatics, as well as the protocol for DNA microarrays on glass slides and data warehousing. Once the microarray data is ready to be dealt with, machine learning methods for microarray data analysis cover main aspects such as clustering, feature selection, classification, data normalization, and missing value imputation. We have also covered the statistical analysis of the data and presented the most popular computer tools to analyze microarray data. Since the use of high-performance computing (HPC) has become very popular in the field, there is a chapter devoted to HPC tools to deal with microarray data. Finally, a chapter discussing the challenges and future trends for microarray analysis closes this book. The book also contains examples and code of research work using microarray data from published articles that are referred to in the references at the end of each chapter. In this way, interested readers can easily find those proposals and results more directly related to each of the subjects addressed in each chapter.

The book is intended for researchers and graduate students in bioinformatics, with basic knowledge in biology and computer science and with a view to work with microarray datasets. The most used tools in this book are R and Weka, both of which can be downloaded free[1,2]. Basic understanding of both is needed to fully take advantage of the presented examples. However, the ideas presented do not assume more than basic knowledge of computer science. We hope our readers enjoy reading this book as much as we enjoyed editing it.

[1] https://www.cs.waikato.ac.nz/~ml/weka/downloading.html

[2] https://www.r-project.org

This book has been made possible thanks to the expert contributors who have carefully put their efforts into writing high-quality chapters about their specific topics. We are grateful to them for making this happen. We are also indebted to the book series editor, John Walker, who specially invited us to edit this book and guided us through the main steps.

*A Coruña, Spain*

*Verónica Bolón-Canedo*
*Amparo Alonso-Betanzos*

# Contents

# Contributors


Giuseppe Agapito • *Department of Medical and Surgical Science, University Magna Graecia, Catanzaro, Italy*

Amparo Alonso-Betanzos • *Research Group LIDIA, Departamento de Computación, CITIC, Universidade da Coruña, A Coruña, Spain*

Dilara Ayyildiz • *Department of Medicine, University of Udine, Udine, Italy*

Verónica Bolón-Canedo • *Research Group LIDIA, Departamento de Computación, CITIC, Universidade da Coruña, A Coruña, Spain*

Ricardo Cao • *Research Group MODES, Department of Mathematics, CITIC and ITMATI, Universidade da Coruña, A Coruña, Spain*

Yi-An Chen • *Laboratory of Bioinformatics, National Institutes of Biomedical Innovation, Health and Nutrition, Ibaraki, Osaka, Japan*

Safaai Deris • *Institute for Artificial Intelligence and Big Data, Universiti Malaysia Kelantan, Kota Bharu, Kelantan, Malaysia*

Ferda Ernawan • *Faculty of Computer Systems & Software Engineering, Universiti Malaysia Pahang, Kuantan, Pahang, Malaysia*

Roberto R. Expósito • *Grupo de Arquitectura de Computadores, CITIC, Universidade da Coruña, A Coruña, Spain*

Kathleen M. Eyster • *Division of Basic Biomedical Sciences, Sanford School of Medicine, University of South Dakota, Vermillion, SD, USA*

Miguel A. Fernández • *Departamento de Estadística e Investigación Operativa, Universidad de Valladolid, Valladolid, Spain*

Oscar Fontenla-Romero • *Computer Science Department, Universidade da Coruña, A Coruña, Spain*

Manuel Franco • *CMN, University of Murcia, Murcia, Spain*

Jorge González-Domínguez • *Grupo de Arquitectura de Computadores, CITIC, Universidade da Coruña, A Coruña, Spain*

Ricardo Gonzalo Sanz • *Statistics and Bioinformatics Unit (UEB), Vall d'Hebron Research Institute (VHIR), Barcelona, Spain*

Mohd Arfian Ismail • *Faculty of Computer Systems & Software Engineering, Universiti Malaysia Pahang, Kuantan, Pahang, Malaysia*

Aws Naser Jaber • *Faculty of Computer Systems & Software Engineering, Universiti Malaysia Pahang, Kuantan, Pahang, Malaysia*

Yolanda Larriba • *Departamento de Estadística e Investigación Operativa, Universidad de Valladolid, Valladolid, Spain*

Ignacio López-de-Ullibarri • *Research Group MODES, Department of Mathematics, CITIC, Universidade da Coruña, A Coruña, Spain*

Kenji Mizuguchi • *Laboratory of Bioinformatics, National Institutes of Biomedical Innovation, Health and Nutrition, Ibaraki, Osaka, Japan*

Mohd Saberi Mohamad • *Institute for Artificial Intelligence and Big Data, Universiti Malaysia Kelantan, Kota Bharu, Kelantan, Malaysia*

Kohbalan Moorthy • *Faculty of Computer Systems & Software Engineering, Universiti Malaysia Pahang, Kuantan, Pahang, Malaysia*

Laura Morán-Fernández • *CITIC, Universidade da Coruña, A Coruña, Spain*

Shyamal D. Peddada • *Department of Biostatistics, University of Pittsburgh, Pittsburgh, PA, USA*

Beatriz Pérez-Sánchez • *Computer Science Department, Universidade da Coruña, A Coruña, Spain*

Silvano Piazza • *Department of Cellular, Computational and Integrative Biology — (CIBIO), University of Trento, Trento, Italy*

Cristina Rueda • *Departamento de Estadística e Investigación Operativa, Universidad de Valladolid, Valladolid, Spain*

Noelia Sánchez-Maroño • *Computer Science Department, CITIC, Universidade da Coruña, A Coruña, Spain*

Alex Sánchez-Pla • *Statistics and Bioinformatics Unit (UEB), Vall d'Hebron Research Institute (VHIR), Barcelona, Spain; Genetics, Microbiology and Statistics Department, University of Barcelona, Barcelona, Spain*

Borja Seijo-Pardo • *CITIC, Universidade da Coruña, A Coruña, Spain*

Lokesh P. Tripathi • *Laboratory of Bioinformatics, National Institutes of Biomedical Innovation, Health and Nutrition, Ibaraki, Osaka, Japan*

Juana-María Vivo • *Department of Statistics and Operations Research, University of Murcia, Murcia, Spain*

# Chapter 1

# Introduction to Bioinformatics

Dilara Ayyildiz and Silvano Piazza

## Abstract

How the scientific community looks at molecular biology today is very different from that 50 years ago. During this time technological developments have led to many significant findings that have shook one of the most important foundations of molecular biology: the central dogma. In this chapter, we will mention how these changes occurred and gave birth to a very important field of today's science, bioinformatics. We will also mention briefly the newest topics of molecular biology regarding bioinformatics technologies and skills.

**Key words** Bioinformatics, High-throughput technology, Molecular biology

## 1 From Molecular Biology to Bioinformatics

Molecular biology has faced a lot of changes in the last decades, comprehending advancements both at technological level and at the level of understanding cell biology. In 1962, Watson and Crick's Nobel Prize in Physiology or Medicine for uncovering the structure and function of DNA was a remarkable milestone in molecular biology, but it also initiated a misleading way of thinking about biological systems. The claim made by Francis Crick (1966) that "The ultimate aim of the modern movement in biology is to explain all biology in terms of physics and chemistry" holds important clues to understand the perspective that was dominant for half a century. The representation of the fact that biological systems are composed of atoms and molecules had opened the way of reductionism in biology. The scientists who believed that biological systems can be explained using the physiochemical aspect of their individual components began to apply their analysis to the putative reduction of Mendelian or population genetics to molecular genetics.

Probably the most famous "poster-child" of reductionism in biology is the central dogma (Fig. 1) known as "DNA makes RNA and RNA makes protein" [1]. As a result of this linear information

Verónica Bolón-Canedo and Amparo Alonso-Betanzos (eds.), *Microarray Bioinformatics*, Methods in Molecular Biology, vol. 1986, https://doi.org/10.1007/978-1-4939-9442-7_1, 

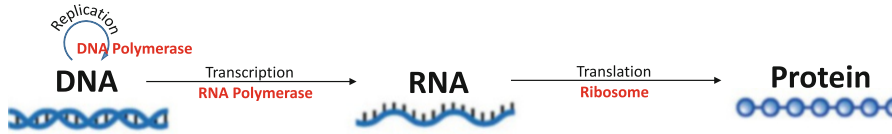


**Fig. 1** Francis Crick's central dogma. The central dogma for biology postulates that the flux of information is going in a one-way direction. Even in this simple form, it had held for several decades, because there was no evidence undermining it

flow, there were certain beliefs about the genome: that it is stationary throughout the life of the organism; it is persistent between cell types and individuals; [2–4] that changes occurring in somatic cells cannot be inherited [5]; and that necessary and sufficient information for cellular function is contained in the gene sequence.

Although the reductionist approach was successful at the beginning of molecular biology, further down the road it was realized that it failed to estimate the complexity of biological systems. The perspective about explaining or predicting an extremely complex and large system by studying their individual parts had to be changed. Today, it is known that a phenomenon of a complex biological system does not result by the specificity of its individual parts only but also by the interaction of these components with each other as well as with other components. The realization of the complexity of biological systems was not new to the scientists, but it acquired a new status after recent technological developments allowed them to simulate these systems using mathematical models [6, 7].

The introduction of computational methods for understanding biological systems has increased the magnitude of the data accumulating in the genomic, transcriptomic, proteomic, and many other omics studies. As a result of the high demand for computational studies, new tools are being developed every day. Hence, the knowledge at each biological level of the central dogma is increasing at an unprecedented pace.

- DNA (genome level). The first organism to have its entire genome sequenced was *Haemophilus influenzae* in 1995, and, at the time of writing this book, several thousands of different organisms had been analyzed, in particular, 1641 animals, 3325 fungi, 679 plants, and 583 protists (Source: NCBI). In addition to this several international ongoing projects are focusing on a specific class of organisms: the 1KITE project to sequence the transcriptome of 1000 insects, the i5k initiative for 5000 arthropods, or the oneKP initiative for more than 1000 plants.
- RNA (transcriptome level). If we consider the transcriptome as the set of all possible RNA molecules of an organism, we actually do not have that definitive answer. In fact, even for humans, we do not know all the possible transcripts because some of them

are To be Experimentally Confirmed (TEC) and maybe some will be discovered in the future. At the moment, 65,000 genes and 224,502 associated transcripts are annotated in the human genome. Moreover, if we consider only the two main public databases for biological high-throughput data (NCBI-GEO and EBI-ArrayExpress) gene expression data are available for more than 3000 different species, and so potentially millions of unique transcripts.

- Protein (proteome level). The same amount of information is available for the protein level: more than 1100 Eukaryota and more than 16,000 organisms have a reference proteome (Source: UniProtKB). But as in other biological layers, a large number of proteins are not analyzed in detail: out of 125 million proteins discovered, only half million are manually annotated and reviewed.

## 2 The Change of Perspective

As a consequence ofthe aforementioned developments the central dogma was revised. The linear information flow from DNA to protein was changed in a more complex way where all the elements of the central dogma would be coordinately working (Fig. 2). The discoveries in genomics, transcriptomics, and proteomics modified the central dogma significantly, and in particular, epigenetics added other important layers to all these fields.

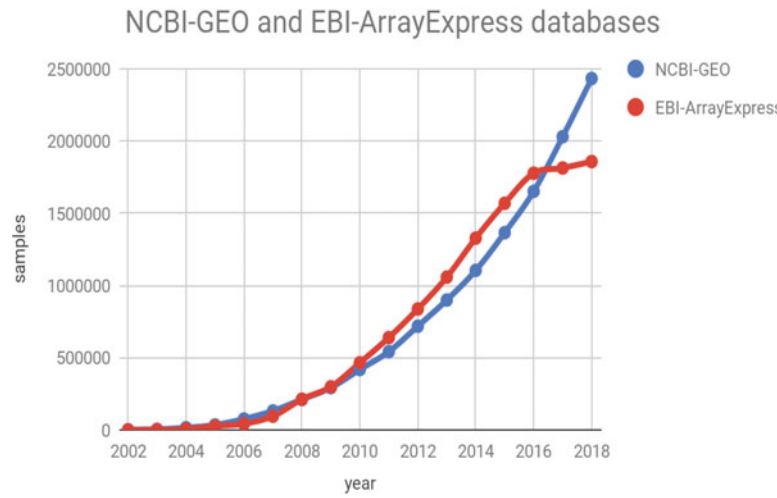


**Fig. 2** Number of samples available in NCI-GEO and EBI-ArrayExpress databases over the last 16 years. The data for the cumulative samples available to analyze were obtained from the corresponding websites

### 2.1 Epigenetics

"Epigenetic," as an adjective of the word "epigenesis," had already been used in the past centuries before we adopted the term epigenetics. In fact, epigenesis referred to the development and conception of fertilized egg giving rise to a complex organism with cells of varied phenotypes. Epigenesis was antagonistic to preformation, according to which the embryo or parts of it are preformed form origination. This reveals the meaning lying behind the word epigenesis: *epi* (over, above) and *genesis* (origin). Similarly, "epigenetics" was used to describe the whole complex of development processes that lie between the genotype and phenotype when the term was first introduced by embryologist Conrad Waddington in 1942 [8]. Waddington's perspective about "epigenotype" was described as "concatenations of processes [are] linked together in a network, so that a disturbance at an early stage may gradually cause more and more far-reaching abnormalities in many different organs and tissues" [8]. He defined a model, called "epigenetic landscape," in order to describe the path followed by a cell. So the possible divergences were determined by the presence or absence of a particular factor.

Years after Waddington, microbiologist David Ledbetter Nanney brought a new concept in "epigenetics" by considering cellular control systems into two genetic systems and auxiliary mechanisms. In his model, by referring to Waddington's paper (1942), he stated that these auxiliary mechanisms (epigenetic) are involved in the determination of specificities to be expressed in a particular cell to emphasize the importance of these systems in genetic systems in cell development [9]. However, research related to regulation of gene expression in development of organisms was not part of epigenetics but molecular genetics.

Following Walther Flemming's discovery of chromosome in 1879, experiments by many investigators, including Wilson and Boveri, provided strong evidence that the developmental program resided in the chromosomes. While Flemming used the term "chromatin" to refer stainable structures in the cell nucleus during cell division, he thought the term would disappear when its chemical composition would be explained. However, it has been used to define the complex of DNA with histone proteins. Studies on chromatin, particularly DNA methylation and histone modifications, started to take place in the 1960s, though they were related to "epigenetics" only after the 1990s.

The chemical modifications of DNA and histone modifications are the result of interactions of DNA with proteins (and RNA) that involve specific enzymes working on specific target sites. Epigenetics did not replace genetic research, but it added another layer of understanding by stating: "This is not the all, there is beyond (*epi*) genetics." In the following paragraphs, we will explore the different methodologies that could be applied.

### 2.1.1 DNA Methylation

The role of DNA methylation in the regulation of gene expression by acting as an epigenetic mark was proposed by Holliday and Riggs in 1975. Then it was shown in vitro experiments that the addition of a methyl group at the 5′ position of cytosine residue (5mC) causes gene inactivation and the pattern of methylation was maintained through the generations of cells [10–13]. Since then, there has been a great focus on DNA methylation profiling for the understanding of the mechanisms particularly on endogenous patterns of DNA methylation, maintenance of the patterns through generations, inactivation of gene expression by methylation, initiation or inhibition of methylation at a fully unmethylated site, discovery of enzymes for de novo methylation, and keeping methylation on already methylated sites. However, the efforts to discover epigenetic modifications was challenged by the lack of high-throughput technologies in the past. Over the last years, increasing demand on these topics has given birth to numerous methods developed to map 5mC, providing insights on genome-wide DNA methylations.

The primitive times of DNA methylation technologies (i.e., pyrosequencing, methylation-specific polymerase chain reaction, and Sanger sequencing) were limited to analyze only the target regions, such as promoter region of a single gene or CpG island. The use of microarray hybridization techniques dramatically increased DNA methylation studies by providing a chance to work at the genome-wide level. Although these studies do not require large amounts of input DNA and money, the high coverage depends on the array design [14]. Thanks to the emergence of next-generation sequencing this is not a concern anymore. It allows for a very detailed analysis of methylation status at the genomic level with high coverage. The mapping of genome-wide DNA methylation can be achieved by the use of chromatin immunoprecipitation (ChIP) with DNA microarray (chip) ChIP-on-chip, or a variation of it called methyl-DNA immunoprecipitation (MeDIP), where purified DNA is immunoprecipitated with an antibody against methylated cytosines, giving rise to genomic maps of DNA methylation. More recently, these techniques have been improved with the use of high-throughput sequencing methodologies, leading to ChIP-seq. Finally, we may have bisulfite sequencing in which the DNA methylation state of cytosines bisulfite may be changed into a methylation-dependent SNP [15]. From the bioinformatics point of view, this last method continues to be the gold standard due to its single-base-pair resolution. A summary of DNA methylation technologies in epigenomics can be seen in Fig. 3.

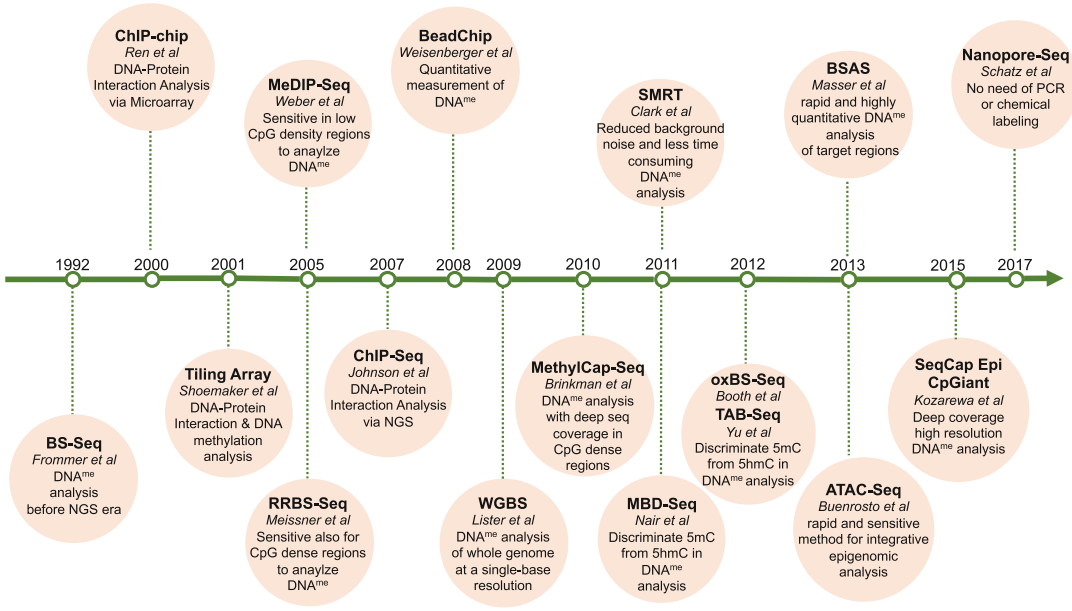


**Fig. 3** Development of high-throughput technologies in epigenomics research with time. *BS-Seq* bisulfite sequencing, *ChIP-chip* chromatin immunoprecipitation with microarray; tiling array, *MeDIP-Seq* methylated DNA immunoprecipitation sequencing, *RRBS-Seq* reduced representation bisulfite sequencing, *ChIP-Seq* chromatin immunoprecipitation sequencing, *BeadChip* bead-based microarray, *WGBS* whole-genome bisulfite sequencing, *MethylCap-Seq* methylation capture sequencing, *MBD-Seq* methyl-CpG binding domain sequencing; sequencing, *SMRT* single-molecule real-time sequencing, *oxBS-Seq* oxidative bisulfite sequencing, *TAB-Seq* TET-associated bisulfite sequencing, *BSAS* bisulfite sequencing, *ATAC-Seq* assay for transposase-accessible chromatin with high-throughput sequencing; SeqCap Epi CpGiant and Nanopore-Seq

### 2.1.2 Histone Modifications

In 1950, Stedman and Stedman [16] assumed that histones were general repressors of gene expression and so different kinds of cells in an organism should have different kinds of histones to create phenotype diversities. So, based on this theory, activation of a gene would require removal of histones; however, the regions that should be detached from histones were not clear. The problem was solved when Allfrey and Mirsky in 1964 [17] reported their findings about gene activation by histone acetylation. Later, other types of histone modifications (i.e., methylation and phosphorylation) were identified. Modern chromatin research was initiated with the discovery of the basic units of eukaryotic chromatin structure called nucleosomes, in 1974, by Kornberg and Thomas [18]. Nucleosomes are formed by 150 bp DNA wrapped around eight histone proteins. After the discovery of nucleosomes, a few years later, Grunstein and his collaborators [19, 20] stated that the histone amino-terminal tails were essential for regulation of gene expression, and for the establishment of silent chromatin domains. In 1996, David Allis [21] identified a histone acetyltransferase enzyme in Tetrahymena similar to a transcriptional regulatory

protein in yeast, suggesting that histone acetylation was related to regulation of gene expression. The same year, another study by Taunton and coworkers provided evidence that a mammalian histone deacetylase was related with transcriptional repressor in yeast. All these studies concluded that both histone modification and nucleosome remodeling were involved in preparing the chromatin template for transcription and, in some cases, the modifications can be transmitted through cell division. In order to create a global map of these modifications, current methodologies are a modified version of ChiP-seq experiments in which DNA fragments of interest are immunoprecipitated by specific histone-modification antibodies.

### 2.2 Epitranscriptomics

Collaborative works in the fields of genomics, transcriptomics, and proteomics together with epigenetics has begun to increasingly elucidate the roles of DNA, RNA, and protein modifications in gene regulation, leading to a new regulatory layer between DNA and protein: "epitranscriptomics." Similar to *epigenomics,* the term itself holds the explanation—study of processes involved in the regulation of transcriptome by RNA molecules themselves. The idea of transcription in the central dogma was changed after the observation that only a small fraction of DNA encodes functional proteins with the release of the human genome sequence [22]. Thereafter, it became clear that the large proportion of the non-protein-coding regions of the genome is also transcribed to produce various RNA regulatory molecules with many different functions [23].

#### 2.2.1 Noncoding RNAs

The regulatory RNA products of these noncoding genes were named according to their base pair lengths: short noncoding RNAs (sncRNA, <200 bp) and long noncoding RNAs (lncRNAs, >200 bp). sncRNAs consist of many different types of RNA molecules such as microRNAs (miRNAs), small interfering RNAs (siRNAs), piwi-interacting RNAs (piRNAs), small nucleolar RNAs (snoRNAs), small nuclear RNAs (snRNAs), extracellular RNAs (exRNAs), small-Cajal body-specific RNAs (scaRNAs), ribosomal RNAs (rRNAs), and transfer RNAs (tRNAs). On the other hand, lncRNAs hold the largest class of noncoding RNAs in the mammalian genome with many different subtypes according to their functions. In addition, they are able to have post-transcriptional modifications such as splicing, 5′-capping, and 3′-polyadenylation [24]. The most known lncRNAs are long intergenic noncoding RNAs (lincRNAs), antisense RNAs (asRNAs), pseudogenes, and circular RNAs (circRNAs).

In order to observe all these new kinds of genes, the suggested experiment is a total RNA extraction followed by sequencing or custom-designed microarrays. Since, usually, they are not expressed at a high level, experiments have to be designed in order to

maximize the low-level spectrum of the signal for microarrays or for RNA-Seq, thus obtaining a high number of reads for each sample.

### 2.2.2 miRNAs and ceRNAs

MicroRNAs (miRNAs) are generally 22 bp long and belong to the sncRNA family. miRNAs are involved in posttranscriptional gene regulation by negatively regulating mRNAs or noncoding transcripts. They are attached to Argonaute proteins to form RNA-induced silencing complexes (RISCs) to be able to bind miRNA-response-elements (MREs) located on the 3′ UTR of mRNAs. As a result of this binding, mRNA degradation or inhibition is achieved to repress the target protein translation [25]. Since a single mRNA can be regulated by a number of miRNAs, they usually contain multiple MREs, which provide binding sites for multiple target miRNAs [26]. The interactions between mRNA and miRNA play a central role in many different biological processes via gene regulation. In this regard, regulation of miRNAs is an important process which occurs via competing for endogenous RNA (ceRNA) interactions [27, 28]. According to ceRNA hypothesis [29], pseudogenes, lncRNAs, and circRNAs compete with mRNAs for the same pool of miRNAs by sharing common MREs and hence acting like miRNA "sponges" [28, 30, 31]. As a result, absorption of miRNA by ceRNA interactions lowers the available level of miRNAs for the target mRNA, thus causing derepression of translation.

In order to observe all these new kinds of genes, the suggested experiment design is a total RNA extraction, followed by size selection to enrich the small fraction of RNAs and then RNA sequencing, or custom-designed microarrays.

### 2.2.3 RNA Methylation

The gradually accumulating studies about ncRNAs have started to change the stable role of RNA in the cell. One of the most striking discoveries in "epitranscriptomics" of course was the modifications of RNA. These processes modulate interactions of RNA with other molecules by affecting RNA structure and hence having a significant impact on cell physiology. RNA methylation is a reversible posttranscriptional modification on RNA, regulating many different pathways, that is, RNA stability and mRNA translation [32–35]. The main types of methylations found in ncRNAs are $N^6$-methyladenosine ($m^6A$), $N^1$-methyladenosine ($m^1A$), 2′-*O*-methylation (2′ OMe/Nm), and 5-methylcytosine ($m^5C$). Among these, $N^6$-methyladenosine is the most abundant modification detected in mRNA [36], but it has also been found in tRNA, rRNA, snRNA, and several lncRNAs [37]. The function of $N^6$-methyladenosine is not fully clear yet; however, it has been shown to have important roles in splicing, stability, nuclear transport, and mediation of cap-dependent translation and turnover by regulating mRNA and ncRNA [38]. On the other hand, methylation at the $N^1$ position of adenosine ($N^1$-methyladenosine) has been shown to

have a role in mRNA translation by close localization to the translation start site and first splice site [36, 39, 40]. The most common RNA modification fundamentally present in rRNAs, snRNAs, tRNAs, and miRNAs is 2′-*O*-methylation [32, 36, 41, 42]. It plays a significant role in protecting against 3′-5′ degradation and 3′ uridylation of some small RNAs in animals and *Drosophila* [32]. Finally, 5-methylcytosine is methylation event that involves the fifth carbon atom of cytosine as a target for methylation in poly (A) RNA, rRNA, tRNA, snRNA, and lncRNA [43, 44]. This type of modification stabilizes tRNA secondary structure and prevents degradation as well as has a role in translation and mRNA transcript stabilization [45].

The technologies for the transcriptome-wide $N^6$-methyladenosine ($m^6A$) profiling came out in 2012, namely, $m^6A$-seq and MeRIP-seq($m^6A$-specific methylated RNA immunoprecipitation sequencing) [37, 46]. These methods allowed researchers to detect $m^6A$ peaks in mRNA, mostly enriched in 3′ UTR regions and long exons. The disadvantages of these methods were the difficulties to mark the actual site of methylation due to the low resolution [47]. As a solution to this problem, UV-based techniques were developed. The combination of $m^6A$-seq with UV-induced RNA–antibody cross-linking gave rise to the technique called photo-cross-linking assisted $m^6A$-seq (PA-$m^6A$-seq). $m^6A$-CLIP (cross-linking immunoprecipitation) [48] and miCLIP ($m^6A$ individual-nucleotide-resolution cross-linking immunoprecipitation) [49] are other UV-based techniques that came out a few years later. On the other hand, some techniques were developed to measure the methylation status ($m^6A$+:$m^6A$ - ratio) of each methylation site. SCARLET (site-specific cleavage and radioactive-labeling followed by ligation-assisted extraction and thin-layer chromatography) [50] and m6A-LAIC-seq ($m^6A$-level and isoform-characterization sequencing) [51] were examples of this category. The aforementioned methods applied for $m^6A$ detection can be used also for $m^6Am$ profiling by adapting them for $m^6Am$ (N6,2′-O-dimethyladenosine). Finally, relatively recent technologies have been developed as single-molecule real-time sequencing by Pacific BioSciences (PacBio) and Nanopore Sequencing by Oxford Nanopore Technologies that could read the property of the molecules, like their modification, directly.

The aforementioned developments in omics sciences and the revision of the central dogma formed the driving force for changing of perspectives in molecular biology. Scientists acquired an opposite of reductionist mindset called systems biology which is based on the collaborative way of thinking to not underestimate the complexity of biological systems. Obviously, studying all different layers of the central dogma individually and also in collaboration needs high-throughput technologies and interpretation of intensive work on

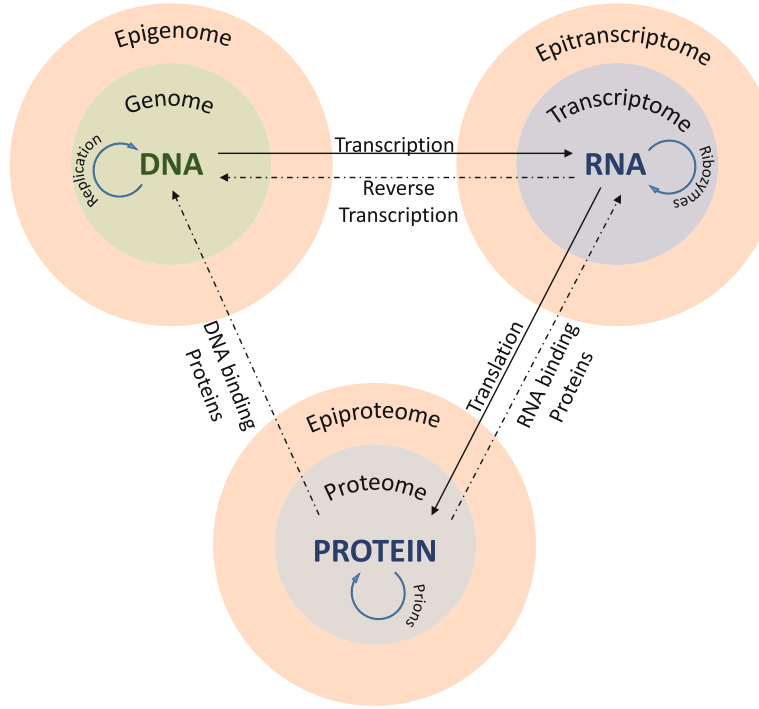


**Fig. 4** Revised central dogma. Graphical representation of the revised biology central dogma. As we can see additional information obtained in the last decade, shown here as the epigenome, the epitrascriptome, and the epiproteome, extend the three canonical layers (DNA, RNA, and protein level). Notably also the flux of information is not more unidirectional but instead creates a complex network

big data science. At this point, the importance of bioinformatics is inevitable in terms of both developing new algorithms and/or applying these tools to provide better insights into systems (Fig. 4).

## 3 Different Levels of Bioinformatics

The different levels of bioinformatics can be divided into three types according to their purposes. The first level is related to the use and improvement of biological databases, organizing and submitting produced data as a result of high-throughput experiments. This task is also known as data curation which holds great importance to make the existing information accessible and updated for the scientific community. This is followed by the usage of these stored data which brings the second task of bioinformatics: developing tools and resources. This level requires an advanced knowledge of computer science which may be difficult to achieve at the beginning of a bioinformatician's career. Many different tools are written in many different programming languages, in relation to

the needs and constant developments in both science and technology. These tools have improved the understanding of biology since it has become possible to analyze big data which is not human readable. The third and last task, integrated closely with biology, is the application of these tools for the interpretation of the results with biological reasoning.

## 3.1 So What Skills or Knowledge Are Needed to Become a Computational Biologist?

### 3.1.1 Personal Skills

Based on what we have discussed in the previous paragraph, we believe that the most important skill is to be able to think from a global perspective of biological systems: in other words, this means to be able to integrate data and analyses trying to understand biological models as a whole with respect to single aspects. This change of perspective, even if it seems a simple advice to follow, is very challenging. In fact, the usual mistake an inexperienced bioinformatician does is trying to confirm with the analyses they perform the theory behind the experiments. But usually, biology is far more complicated than expected. For example, let us assume that you are involved in a project in which a drug is a promising candidate of a high-throughput functional screening and the biological readout for the experiment is the reduced cell mobility. As a second step of the project, the team decides to perform a global transcriptional profile of mRNA.

In this situation, it is very easy to try to focus on the possible pathways linked to cell motility, cell migration, or adhesion. Instead, rigorous analyses may provide unexpected results: for instance, the drug causes bioenergetics impairment interfering with mitochondrial activity, leading to a modified cell proliferation rate that in the end is responsible for the differential cell mobility [52].

A second, maybe unexpected, skill to have is patience and proceed at a slow pace with the analyses. This is for two different reasons: (1) Most of the time you will spend your days writing scripts, or compiling or installing specific programs. The advice that we want to give you is to take your time to perform these tasks and comment your code in order to be able to review your work in the future and to avoid simple mistakes. Most of the time you will perform the analyses more than once or you want to utilize the code you have written for similar projects. Without a proper annotation, you will not be able to remember all the specific steps you choose and why you do so. Another related advice is that you may try to write the detailed method description as you perform the analysis. In this way, you will have an explanatory document ready to share with your colleagues and collaborators, or to be included in a manuscript. In the long run, you will obtain results faster because you become more efficient. (2) A very important good practice is to read the documentation that often comes with the software you are planning to use. Once again, do not rush into your analyses, take your time and try to reproduce with the sample data

the expected results. When you have become familiar with the tools, you may feel confident to try with your data.

In bioinformatics analyses, almost always the software is designed with internal methods of quality control which employs statistical methods to monitor and control the robustness of the results. For example, if a statistical test is applied in a high-throughput experiment procedure, the multiple test correction is automatically computed (see the other chapters in the book for real examples). So you may feel confident that all the statistical controls are already done and your job is more related to the interpretation of results. Even though this may be true in some cases, usually it is not enough. In fact, it is your duty to check if the model may apply or not to your specific case; for example, a common mistake is using parametric statistical tests when the sample size is small, or in the presence of obviously violated assumptions of the model. In fact, as a computational biologist, you will constantly struggle with tool design for applications such as psychology, business analytics, or economics where the sample size is not a problem in respect to biology where if you have a triplicate you are lucky. So, in the end, you have to be very careful handling data and always consider the quality and quantity of our input data, perform the quality checks on the intermediate data, and test similar tools in order to check the consistency of the results. In other words, try to use the same strategy that is commonly used at the bench in the wet laboratory, that is, when you need to prove an idea, several experiments with a common biological readout are performed.

### 3.1.2 Programming Skills

In terms of specific programming skills, a common question is which language is the best for learning bioinformatics. Since this is a very general question, the answer is also very generic and not very satisfactory: it depends on your specific application. For example, if your interest is writing programs where the calculation speed in a high-performance framework is the most critical parameter, an obvious choice would be a low-level language such as C, C++, GO, or Swift. On the other hand, if you would like to build up software with a graphical interface, Java may be an optimal choice. Finally, if you are more interested in data processing and great visualization capability, you may use Python or R. But in all cases, the most important factors that may help you to decide are not the intrinsic property of the languages but their extrinsic one: the available libraries related to your task. When you perform any kind of analyses, you will be surprised that most of the times, other people already have encountered the same challenge. So probably you may want to take advantage of others' experience to quickly find solutions to your problem. In other words, you do not want to waste time reinventing the wheel. In bioinformatics this means that specific extensions may be available to your software. In Python, they are called python modules, and in R language, they are called R

packages. Moreover, some of them are also organized in specific biologically related collections as Biopython or R/Bioconductor. At the time of writing this book, there are several thousand extensions available for each language, covering the majority of bioinformatics needs you may encounter. But the most interesting part of these projects is that they are open source, meaning that you have access to the source code used for their implementations. For medium to advance users this means to have a template of a complex statistical package and in this way be able to learn how to create their own.

### 3.1.3 Time to Say Hello to Linux

The modern operating systems like Windows 10 or OSX are wonderful tools that you may use as your primary setup for bioinformatics analyses. The majority of the software we discussed so far have an implementation for them. So you may think that those old days with programs that must be operated by text-only interface running in a Unix/Linux machine are gone for good. Well, that is not actually the case.

In our bioinformatics day-to-day work, we always use Unix/Linux machines for one simple reason: scalability. As long as you will continue to work in bioinformatics, you will find yourself involved in big projects with, for example, thousands of samples with high-throughput data that would require a huge amount of computational power. Even modern top-of-the-line laptops or even workstations are not capable enough to handle such heavy-duty tasks. So the use of high-performance-computing (HPC) platform is nowadays an essential requirement, and usually, all these data centers are operated by Unix/Linux systems. So, in order to avoid reproducibility problems or software incompatibility issues, it is a good idea to have the same system in your daily machine (the test set) and in the HPC (the analysis set). Fortunately, that means that you do not need to become a Linux guru: probably you may just need to become familiar with the use of the Linux shell (e.g., bash) and with some text-manipulation programs like sed or awk. In fact, very often, you will encounter tasks that are merely preprocessing steps because the input files are not directly readable. An apparent simpler approach in those situations may seem to be the use of Microsoft Excel. But remember that every time, in your workflow, there is a step that could not be automated by some scripts, you are taking the chance to have errors difficult to debug in the future, or to obtain a procedure irreproducible by others.

Nevertheless, you do not need to worry, since nowadays several available distributions are very easy to install, maintain, and use. Just to cite some examples of what you may encounter, we can mention Scientific Linux, Bio-Linux, or CentOS distributions.

Another smart possibility may be installing your favorite Linux flavor into a virtualized or dockerized environment. Both solutions

rely on a simple principle: your machine becomes the host of another virtualized operating system; in this way you may continue to use your Windows or OSX for your daily activities and the Linux installation for the bioinformatics analyses. On modern computers with at least 32GB of RAM memory, it is not a problem to utilize this kind of setup without any degradation of performance.

## References


1. Crick F (1970) Central dogma of molecular biology. Nature 227:561–563
2. Pelak K, Shianna KV, Ge D et al (2010) The characterization of twenty sequenced human genomes. PLoS Genet 6(9):e1001111
3. Roach JC, Glusman G, Smit AF et al (2010) Analysis of genetic inheritance in a family quartet by whole-genome sequencing. Science 328:636–639
4. Durbin RM, Abecasis GR, Altshuler DL et al (2010) A map of human genome variation from population-scale sequencing. Nature 467:1061–1073
5. McDaniell R, Lee BK, Song L et al (2010) Heritable individual-specific and allele-specific chromatin signatures in humans. Science 328:235–239
6. Alm E, Arkin AP (2003) Biological networks. Curr Opin Struct Biol 13:193–202
7. Emmeche C (1997) Aspects of complexity in life and science. Philosophica 59:41–68
8. Waddington CH (1942) The epigenotype. Endeavor 1:18–20
9. Nanney DL (1958) Epigenetic control systems. PNAS 44:712–717
10. Naveh-Many T, Cedar H (1981) Active gene sequences are undermethylated. PNAS USA 78:4246–4250
11. Stein R, Gruenbaum Y, Pollack Y et al (1982a) Clonal inheritance of the pattern DNA methylation in mouse cells. Proc Natl Acad Sci U S A 79:61–65
12. Stein R, Razin A, Cedar H (1982b) In vitro methylation of the hamster adenine phosphoribosyltransferase gene inhibits its expression in mouse L cells. Proc Natl Acad Sci U S A 79:3418–3422
13. Deichmann U (2014) Interview with Howard Cedar, 19 June 2014 (Jerusalem, authorized transcript)
14. Yong WS, F-Hsu M, Chen PY (2016) Profiling genome-wide DNA methylation. Epigenetics Chromatin 9:26
15. Hajkova P, el-Maarri O, Engemann S et al (2016) DNA-methylation analysis by the bisulfite-assisted genomic sequencing method. Methods Mol Biol 200:143–154
16. Stedman E, Stedman E (1950) Cell specificity of histones. Nature 166:780–781
17. Allfrey VG, Mirsky AE (1964) Acetylation and methylation of histones and their possible role in the regulation of RNA synthesis. Proc Natl Acad Sci 51:786–794
18. Kornberg RD, Thomas JO (1974) Chromatin structure; oligomers of histones. Science 184 (4139):865–868
19. Wallis JW, Hereford L, Grunstein M (1980) Histone H2B genes of yeast encode two different proteins. Cell 22:799–805
20. Durrin LK, Mann RK, Kayne PS et al (1991) Yeast histone H4 N-terminal sequence is required for promoter activation in vivo. Cell 65:1023–1031
21. Allis CD, Glover CV, Gorovsky MA (1979) Micronuclei of Tetrahymena contain two types of histone H3. Proc Natl Acad Sci U S A 76:4857–4861
22. Venter JC, Adams MD, Myers EW et al (2001) The sequence of the human genome. Science 291:1304–1351
23. ENCODE Project Consortium, Birney E, Stamatoyannopoulos JA et al (2007) Identification and analysis of functional elements in 1% of the human genome by the ENCODE pilot project. Nature 447:799–816
24. Quinn JJ, Chang HY (2016) Unique features of long non-coding RNA biogenesis and function. Nat Rev Genet 17:47–62
25. Bartel DP (2009) MicroRNAs: target recognition and regulatory functions. Cell 136 (2):215–233
26. Pasquinelli AE (2012) MicroRNAs and their targets: recognition, regulation and an emerging reciprocal relationship. Nat Rev Genet 13(4):271–282
27. Tay Y, Salmena L, Weiss D et al (2011) Coding-independent regulation of the tumor suppressor PTEN by competing endogenous mRNAs. Cell 147(2):344–357
28. Cesana M, Cacchiarelli D, Legnini I et al (2011) A long noncoding RNA controls

muscle differentiation by functioning as a competing endogenous RNA. Cell 147 (2):358–369

29. Salmena L, Poliseno L, Tay Y et al (2011) A ceRNA hypothesis: the Rosetta Stone of a hidden RNA language? Cell 146(3):353–358
30. Poliseno L, Salmena L, Zhang J et al (2010) A coding-independent function of gene and pseudogene mRNAs regulates tumour biology. Nature 465(7301):1033–1038
31. Ebert MS, Sharp PA (2010) MicroRNA sponges: progress and possibilities. RNA 16 (11):2043–2050
32. Ji L, Chen X (2012) Regulation of small RNA stability: methylation and beyond. Cell Res 22:624–636
33. Wang X, Lu Z, Gomez A et al (2014) N6-methyladenosine-dependent regulation of messenger RNA stability. Nature 505:117–120
34. Wang X, Zhao BS, Roundtree IA et al (2015) $N^6$-methyladenosine modulates messenger RNA translation efficiency. Cell 161:1388–1399
35. Dev RR, Ganji R, Singh SP et al (2017) Cytosine methylation by DNMT2 facilitates stability and survival of HIV-1 RNA in the host cell during infection. Biochem J 474:2009–2026
36. Roundtree IA, Evans ME, Pan T et al (2017) Dynamic RNA modifications in gene expression regulation. Cell 169:1187–1200
37. Dominissini D, Moshitch-Moshkovitz S, Schwartz S et al (2012) Topology of the human and mouse m6A RNA methylomes revealed by m6A-seq. Nature 485:201–206
38. Meyer KD, Patil DP, Zhou J et al (2015) 5′ UTR m(6)A promotes cap-independent translation. Cell 163:999–1010
39. Dominissini D, Nachtergaele S, Moshitch-Moshkovitz S et al (2016) The dynamic N(1)-methyladenosine methylome in eukaryotic messenger RNA. Nature 530:441–446
40. Li X, Xiong X, Wang K et al (2016) Transcriptome-wide mapping reveals reversible and dynamic N(1)-methyladenosine methylome. Nat Chem Biol 12:311–316
41. Schibler U, Perry RP (1977) The 5′-termini of heterogeneous nuclear RNA: a comparison among molecules of different sizes and ages. Nucleic Acids Res 4:4133–4149
42. Borges F, Martienssen RA (2015) The expanding world of small RNAs in plants. Nat Rev Mol Cell Biol 16:727–741
43. Amort T, Souliere MF, Wille A et al (2013) Long non-coding RNAs as targets for cytosine methylation. RNA Biol 10:1003–1008
44. Lewis CJ, Pan T, Kalsotra A (2017) RNA modifications and structures cooperate to guide RNA-protein interactions. Nat Rev Mol Cell Biol 18:202–210
45. Esteller M, Pandolfi PP (2017) The epitranscriptome of noncoding RNAs in cancer. Cancer Discov 7:359–368
46. Meyer KD, Saletore Y, Zumbo P et al (2012) Comprehensive analysis of mRNA methylation reveals enrichment in 3′ UTRs and near stop codons. Cell 149:1635–1646
47. Schwartz S, Agarwala SD, Mumbach MR et al (2013) High-resolution mapping reveals a conserved, widespread, dynamic mRNA methylation program in yeast meiosis. Cell 155 (6):1409–1421
48. Ke S, Alemu EA, Mertens C et al (2015) A majority of m6A residues are in the last exons, allowing the potential for 3′ UTR regulation. Genes Dev 29(19):2037–2053
49. Linder B, Grozhik AV, Olarerin-George AO et al (2015) Single-nucleotide-resolution mapping of m6A and m6Am throughout the transcriptome. Nat Methods 12(8):767–772
50. Liu N, Parisien M, Dai Q et al (2013) Probing N6-methyladenosine RNA modification status at single nucleotide resolution in mRNA and long noncoding RNA. RNA 19 (12):1848–1856
51. Molinie B, Wang J, Lim KS et al (2016) m6A-LAIC-seq reveals the census and complexity of the m6A epitranscriptome. Nat Methods 13(8):692–698
52. Kalyanaraman B, Cheng G, Hardy M et al (2018) A review of the basics of mitochondrial bioenergetics, metabolism, and related signalling pathways in cancer cells: Therapeutic targeting of tumor mitochondria with lipophilic cationic compounds. Redox Biol 14:316–327

# Chapter 2

## Protocol for DNA Microarrays on Glass Slides

Kathleen M. Eyster

### Abstract

The DNA microarray is a powerful, flexible, nonbiased discovery technology. Microarrays can be used to assess processes from gene expression to long noncoding RNAs to specific pathologies, as well as many others. This chapter describes the protocol for DNA microarray analysis of differential gene expression using DNA sequences spotted on microscope slides.

**Key words** DNA microarray, Gene expression, RNA extraction, RNA analysis

### 1 Introduction

The microarray is a flexible, nonbiased technology that can be used to assess a variety of entities. Early arrays were designed to analyze gene expression using the now-familiar grid pattern of DNA sequences attached to a platform such as a glass slide or specialized cassette [1]. Over time, microarrays have been designed to assess DNA mutations [2], DNA methylation [3], including single nucleotide polymorphisms [4], chromosomal fragments [5], microRNAs [6], and long noncoding RNAs [7]. The technique of chromatin immunoprecipitation can be combined with DNA microarray technology [8]. Microarrays have been designed to analyze peptides [9], tissue expression of specific proteins [10], and other factors [11]. Some DNA microarrays cover entire genomes [1, 12], whereas others are focused on a subset of genes that have been shown to be involved in a specific function or pathology such as foodborne viruses [13] or apoptosis [14], and custom microarrays can be designed [15]. Microarrays were initially focused on basic research, but applications have entered clinical practice and may be used as the basis for clinical diagnosis or clinical decision making [4, 16], and microarrays are applied in agriculture [17, 18].

This chapter describes the performance of DNA microarray experiments using DNA sequences attached to treated microscope

Verónica Bolón-Canedo and Amparo Alonso-Betanzos (eds.), *Microarray Bioinformatics*, Methods in Molecular Biology, vol. 1986, https://doi.org/10.1007/978-1-4939-9442-7_2, 

slides for the analysis of differential gene expression. The protocol involves extraction and purification of total RNA, processing of RNA through synthesis of double-stranded cDNA, synthesis of labeled antisense RNA, hybridization with the DNA microarray, and posthybridization processing and scanning of the microarrays.

## 2 Materials

1. RNAlater (Ambion) or similar RNA preservation reagent.
2. Tri reagent (Molecular Research Center).
3. 1.5 mL and 2 mL microfuge tubes.
4. Polytron homogenizer with 7 mm probe (Kinematica) for processing tissue, or pellet pestle with cordless motor (Kimble Kontes) or glass Dounce homogenizer for processing cultured cells (Kimble Kontes).
5. Bromochloropropane.
6. Sodium acetate, 3 M.
7. RNase Zap (Ambion) or RNase Away (Life Technologies).
8. Diethylpyrocarbonate (DEPC)-treated water: Add 1 mL of DEPC (Sigma) to 999 mL purified water (18.2 MΩ resistivity) and stir on a stir plate overnight. Destroy the toxicity of DEPC by autoclaving the water. Store at room temperature.
9. Silica-based RNA purification spin columns such as the RNeasy kit (Qiagen) or equivalent. The RNeasy Kit contains silica-based spin columns, binding and washing buffers, and nuclease-free water.
10. RLT buffer (Qiagen RNeasy kit): Requires addition of 10 μL β-mercaptoethanol to 1 mL of the RNeasy lysis buffer/binding RLT buffer before use.
11. 100% ethanol, molecular grade.
12. RPE buffer (Qiagen RNeasy kit): Requires addition of 44 mL molecular grade 100% ethanol to 11 mL washing buffer RPE concentrate before use.
13. RNase-free DNase (Qiagen) stock solution: Gently mix 550 μL nuclease-free water with the vial of lyophilized DNase.
14. RNase-free DNase (Qiagen) working solution: Gently mix 10 μL DNase stock solution with 70 μL RNase-free DNase dilution buffer for each sample.
15. Agilent Bioanalyzer and Nano 6000 LabChip and reagents for RNA analysis.
16. Preparation of gel matrix for the Agilent RNA 6000 Nano Lab Chip: Add 550 μL of gel matrix to a spin filter cartridge

mounted in a microfuge tube. Centrifuge the gel at 1500 × *g* for 10 min at room temperature. The filtered gel can be stored in 65 µL aliquots at 4 °C for up to 1 month.

17. MessageAmp II-Biotin *Enhanced* Kit (Ambion): contains T7 oligo dT primer, ArrayScript reverse transcriptase, RNase inhibitor, 10× first strand buffer, dNTP mix, 10× second strand buffer, DNA polymerase, RNase H, T7 enzyme mix, T7 10× reaction buffer, biotin-NTP mix, nuclease-free water, cDNA filter cartridges, and aRNA filter cartridges.
18. Reverse Transcription Master Mix: Mix 1 µL nuclease-free water, 1 µL T7 oligo dT primer, 2 µL 10× first strand buffer, 4 µL dNTP mix, 1 µL RNase inhibitor, and 1 µL ArrayScript reverse transcriptase for each sample. Increase the volume by 5% to account for pipetting overage.
19. Second Strand Master Mix: Mix 63 µL nuclease-free water, 10 µL second strand buffer, 4 µL dNTP mix, 2 µL DNA polymerase, and 1 µL RNase H for each sample plus 5% volume overage.
20. Wash buffer for cDNA and aRNA spin column purification: add 24 mL ethanol to the entire bottle of Wash buffer before using.
21. In vitro Transcription (IVT) Master Mix: Mix 4 µL T7 10× reaction buffer, 4 µL T7 enzyme mix, and 12 µL Biotin-NTP mix for each sample plus 5% volume overage. (The Biotin-NTP mix contains ATP, GTP, CTP, and biotin-11-UTP.)
22. DNA microarrays spotted on treated microscope slides, such as those from Microarrays, Inc., Phalanx Biotech, or CodeLink.
23. Flexible adhesive cover slip gaskets (HybriWell, Grace Bio-Labs, Inc) with sealing strips.
24. Prehybridization buffer: 5× SSC, 0.1% Tween 20, 0.1% bovine serum albumin (BSA), filtered through 0.2 µm filter. Store frozen at −20 °C (*see* **Note 1**).
25. Hybridization buffer: 40–70% deionized formamide, 10× SSC, 0.2% SDS, 0.02% sheared salmon sperm DNA, or purchased hybridization buffer designed for microarrays.
26. Posthybridization wash buffer #1: 0.075 M Tris, pH 7.6, 0.1125 M NaCl, 0.0375% Tween 20. Filter through 0.2 µm filter and store at room temperature.
27. Posthybridization wash buffer #2: 0.1 M Tris, pH 7.6, 0.15 M NaCl, 0.05% Tween 20. Filter through 0.2 µm filter and store at room temperature.
28. Posthybridization wash buffer #3: 0.1× SSC, 0.05% Tween 20. Filter through 0.2 µm filter and store at room temperature.

29. TNB buffer: 0.1 M Tris–HCl, pH 7.6, 0.15 M NaCl, 0.5% NEN blocking reagent (Perkin Elmer). Mix liquid buffer components and heat to 60 °C on stir plate. Add NEN blocking reagent slowly to warm buffer. Stir overnight. Filter through 0.88 μm filter and store in 50 mL aliquots at −20 °C.
30. Streptavidin Alexa 647 stock solution: Mix 1 mL of 1× PBS, pH 7.4, with 1 mg of Streptavidin Alexa 647 Fluor (Molecular Probes, now Thermofisher). Store aliquots of the stock solution at −80 °C in an opaque box.
31. Streptavidin Alexa 647 working dilution: Mix in the ratio of 10 μL Streptavidin Alexa 647 stock solution to 4990 μL TNB buffer in a volume sufficient to submerge the slides in the fluorescent dye in the chosen reservoir.

## 3 Methods

### 3.1 RNA Hygiene

Technologies such as DNA microarray and reverse transcription-quantitative polymerase chain reaction (RT-qPCR) require very high-quality purified RNA. The ubiquity of RNases requires that steps must be taken to prevent RNA degradation. It is also critical to remove substances such as genomic DNA that may contaminate the RNA.

1. Clean all of the hard surfaces of instruments, bench tops, racks, and any other materials that will be used in the processing of tissues or cells for extraction of RNA with a reagent that destroys RNases such as RNase Zap. Rinse with DEPC-treated water.
2. Wear gloves at all times and restrain long hair to avoid contamination with RNases from the skin and hair. Change gloves often.
3. Destroy contaminating RNases in water by treatment with diethylpyrocarbonate (DEPC) as described in Subheading 2, **item 8**.
4. Laboratory disposables such as microfuge tubes and pipette tips should be nuclease-free.

### 3.2 Extraction of RNA from Cultured Cells

1. Remove culture medium from the cell cultures by centrifugation of cells growing in suspension; resuspend the cells in saline to rinse and centrifuge again. For adherent cells, rinse with saline.
2. Add 1 mL Tri reagent to pelleted cells (grown in suspension) or to the culture dish (for adherent cells) (*see* **Notes 2** and **3**).
3. Pipet the Tri reagent/cell debris into a 1.5 mL microfuge tube.

4. Disperse the cellular contents equally throughout the Tri reagent with an instrument such as the pellet pestle and cordless motor or a Dounce glass homogenizer.

### 3.3 Extraction of RNA from Tissues

1. At the time of collection, store dissected tissues in an RNA preservation reagent such as RNAlater (*see* **Notes 4** and **5**) and store at −80 °C.
2. Thaw tissue, blot to remove RNAlater, and weigh a tissue piece no larger than 30 mg.
3. Mince the tissue using scalpels or scissors under Tri reagent.
4. Place the minced tissue in 1 mL of Tri reagent in a 12 × 75 mm nuclease-free polypropylene test tube.
5. Use a Polytron instrument with a 7 mm probe or equivalent to homogenize the tissue (*see* **Notes 6–8**), pulsing for 10 s three times with a 30-s rest period on ice between pulses (*see* **Note 9**).
6. After homogenizing each sample, rinse the probe with DEPC-treated water, spray with RNase Zap, rinse again with DEPC-treated water, run the probe in a graduated cylinder filled with DEPC-treated water, and dry the probe in preparation for the next sample.
7. Homogenize all samples then thoroughly clean the probe. The corrosiveness of Tri reagent will ruin the probe if not removed as quickly as possible (*see* **Note 10**).

### 3.4 Purification of RNA

1. To each 1 mL of cultured cells dispersed in Tri reagent or sample of tissue homogenized in 1 mL of Tri reagent, add 60 μL of 3 M sodium acetate and 200 μL of bromochloropropane. Shake the samples vigorously to mix while holding the caps on tightly to prevent spillage. Fresh Tri reagent is a clear pink; it will turn milky with this mixing step. Incubate on ice for 15 min.
2. Separate the phases by centrifugation for 5 min at 8000 × *g*.
3. After centrifugation, transfer the aqueous layer (the top clear layer) to a nuclease-free 12 × 75 mm test tube (*see* **Note 11**).
4. Add 1 mL RLT buffer and 1.2 mL 100% ethanol to the RNA sample (*see* **Note 12**).
5. Place a spin column with a silica-based membrane into a microfuge tube and dispense 700 μL of the RNA sample into the column. Centrifuge for 30 s at 10,000 × *g* and discard the eluent.
6. Repeat the centrifugation of 700 μL aliquots until all of the sample has been filtered through the spin column.

7. Using wash buffer RW1 (from the extraction kit), add 350 μL of the RW1 buffer to the column. Centrifuge for 30 s at 10,000 × *g* and discard the eluent.
8. Carry out a DNase treatment on the column by adding 80 μL RNase-free DNase working reagent directly to the membrane of the column.
9. Allow the DNase to act for 15 min at room temperature.
10. Wash the column again with 350 μL wash buffer RW1; centrifuge the column at 10,000 × *g* for 30 s and discard the eluent.
11. Wash the column twice with buffer RPE (from the extraction kit). For each wash, add 500 μL RPE to the column, centrifuge for 30 s at 10,000 × *g*, and discard the eluent.
12. Transfer the column to a clean, dry microfuge tube. Centrifuge the column for 2 min at 14,000 × *g* without adding any buffer to dry the column.
13. Transfer the column to a fresh, nuclease-free 1.5 mL microfuge tube. Ensure that it is well labeled as this will be the collection and storage tube for the purified RNA sample.
14. Deliver 50 μL of nuclease-free water to the center of the column and incubate at room temperature for 10 min (*see* **Note 13**).
15. Centrifuge the column at 10,000 × *g* for 1 min to elute the RNA. Do not discard the eluent; it contains the purified RNA.
16. Deliver a second aliquot of 50 μL of nuclease-free water to the center of the column. Incubate at room temperature for 10 min.
17. To elute the second aliquot of RNA into the microfuge collection tube containing the first elution volume, centrifuge the column at 10,000 × *g* for 1 min (*see* **Note 14**).
18. Remove the column from the microfuge collection tube and discard.
19. Save a 3–5 μL aliquot of the RNA separately to use for analysis of the quantity and purity of the RNA.
20. Store the purified RNA sample and the aliquot for analysis at −80 °C until used for DNA microarray or other downstream applications.

### 3.5 Analysis of the Purified RNA

Use the Agilent RNA 6000 Nano Lab Chip to analyze the quantity and quality of the purified RNA (*see* **Notes 15** and **16**).

1. The kit reagents of marker mixture, dye concentrate, and one 65 μL aliquot of prepared gel matrix should be removed from storage at 4 °C 30 min before use. Place them in an opaque box to come to room temperature in the dark.

2. Turn on water bath to come to temperature at 70 °C.
3. Vortex the dye concentrate well, then centrifuge for 5–10 s at 10,000 × *g*.
4. Add 1 μL of the dye concentrate to the 65 μL aliquot of prepared gel matrix and vortex thoroughly to mix. Centrifuge at 13,000 × *g* for 10 min.
5. Thaw the 3–5 μL aliquots of RNA samples that had been set aside for analysis as well as the RNA ladder.
6. Heat the RNA samples and RNA ladder at 70 °C for 2 min in the water bath to denature them. Chill all samples on ice. When the samples are chilled, centrifuge for 1 min at 10,000 × *g* to return the condensate to the bottom of the tubes.
7. Add the gel matrix to the appropriate well on the chip as designated by the manufacturer and pressurize the gel. Add the gel, marker mixture, RNA ladder, and RNA samples to the appropriate wells.
8. Using a vortexer with speed adjustment and an adaptor, vortex the chip at 2400 rpm for 1 min.
9. Place the chip in the Agilent BioAnalyzer and follow the prompts of the software to read the chip. Record the RIN number and RNA concentration for each sample (*see* **Note 17**).
10. Remove the chip from the machine and clean the electrodes with nuclease-free water as soon as the chip is finished reading to avoid undue corrosion of the electrodes.

### 3.6 Preparation of Antisense RNA

The next series of reactions is designed to synthesize biotinylated antisense RNA (aRNA, alias complementary RNA or cRNA) in preparation for hybridization.

1. The quantity of total RNA required for the microarray platform that you are using will be designated by the manufacturer. The desired quantity is usually 0.2–2.0 μg total RNA in 10 μL volume. Calculate the volume containing the proper quantity of RNA for each sample. Bring the volume to 10 μL with nuclease-free water, or use a vacuum concentrator such as a SpeedVac to reduce the volume to 10 μL as necessary (*see* **Note 18**).
2. To each sample of total RNA, add reverse transcription master mix (10 μL).
3. Incubate the samples in a well-controlled heat block or thermocycler for 2 h at 42 °C. Place on ice to cool after the incubation. The first strand of cDNA has now been synthesized.

4. Place vials of nuclease-free water (24 μL per sample) on a heat block or in a water bath and bring to 55 °C in preparation for the cDNA purification step later.
5. To each sample, add 80 μL of second strand master mix.
6. Incubate the samples at 16 °C for 2 h in a well-controlled heat block or thermocycler to synthesize the second strand of cDNA. Cool the samples on ice after the incubation.
7. Purify the resulting double-stranded cDNA using the cDNA filter cartridges. To each sample of double-stranded cDNA, add 250 μL of cDNA binding buffer to yield a total volume of 350 μL; pipet up and down to mix.
8. Transfer the cDNA samples to the filter cartridges.
9. Centrifuge the filter cartridges for 1 min at 10,000 × *g*.
10. Discard the eluent and wash the filter cartridge with 500 μL of wash buffer; add the wash buffer and centrifuge for 1 min at 10,000 × *g*. Discard the eluent of the wash buffer.
11. Dry the filter cartridges by centrifugation for 1 min at 10,000 × *g* in the absence of added buffer.
12. Move the filter cartridge to a new, nuclease-free, carefully labeled collection tube.
13. To the center of each filter cartridge, add 22 μL of preheated 55 °C nuclease-free water.
14. Leave on the bench at room temperature for 2 min.
15. Elute the double-stranded cDNA into the collection tubes by centrifugation for 1 min at 10,000 × *g*.
16. To each 20 μL sample of purified double-stranded cDNA, add 20 μL of IVT master mix.
17. Carry out the IVT reaction in a well-controlled heat block or thermocycler for 14 h at 37 °C. Biotin-labeled aRNA is synthesized in this reaction (*see* **Note 19**).
18. Stop the IVT reaction by adding 60 μL room temperature nuclease-free water to each sample.
19. Bring nuclease-free water (200 μL per sample) to 55 °C in a heat block or water bath in preparation for the aRNA purification step (*see* **Note 20**).
20. Transfer 350 μL of the aRNA binding buffer to each sample of synthesized aRNA.
21. Pipet 250 μL of 100% ethanol into the first sample and gently pipet up and down to mix. Do not vortex or centrifuge as this will cause the aRNA to precipitate and will reduce recovery of the sample. As soon as the reagents are mixed, transfer the entire volume to an aRNA filter cartridge in a collection tube. Complete the addition of ethanol, gentle mixing, and transfer

to the filter cartridge for each sample before moving on the next sample.

22. Centrifuge the samples for 1 min at 10,000 × *g* and discard the eluent.
23. Add 650 μL wash buffer to each filter cartridge and centrifuge for 1 min at 10,000 × *g* to wash the filter, and discard the eluent.
24. Without adding buffer, centrifuge the filter cartridges for 1 min at 10,000 × *g* to dry the filters.
25. Transfer the filter cartridges to well-labeled, new, nuclease-free collection tubes.
26. Transfer 200 μL of 55 °C nuclease-free water to each filter cartridge. Place the samples in the 55 °C water bath or heat block to incubate for 10 min.
27. Centrifuge the filter cartridges for 90 s at 10,000 × *g* to elute the purified biotinylated aRNA (*see* **Note 21**).
28. The concentration of aRNA in each sample can be calculated by reading the absorbance at a wavelength of 260 nm ($A_{260}$) in a spectrophotometer (*see* **Note 22**). Use the equation:

$$A_{260} \times \text{dilution factor} \times 40\ \mu g/mL \times 0.001\ mL/\mu L = \mu g\ aRNA/\mu L$$

29. Aliquot the aRNA into aliquots containing 10 μg aRNA each and freeze at −80 °C, or proceed directly to hybridization.

### 3.7 Hybridization

Preparation for hybridization requires final preparation of the samples and prehybridization of the microarray slides. These two steps should be carried out simultaneously so that the prehybridized slides are ready at the same time as the samples.

1. Turn on the heat block and set to 90 °C for the denaturation reaction. Make sure the prehybridization buffer has been warmed to 55 °C. Turn on the shaking incubator or hybridization oven and warm to 42 °C in preparation for the hybridization reaction.
2. For sample preparation, compute the volume of each aRNA sample that contains 10 μg of aRNA. Raise the volume to 20 μL with nuclease-free water or reduce the volume to 20 μL using a vacuum concentrator as necessary.
3. Pipet 5 μL of fragmentation buffer into the 20 μL sample of biotinylated aRNA and heat at 94 °C for 20 min in a thermocycler or heat block (*see* **Note 23**).
4. Place on ice to cool for at least 5 min.
5. Centrifuge for 30 s at 10,000 × *g* to return condensate to the bottom of the tubes.

6. Transfer 185 μL of hybridization buffer to the 25 μL of fragmented aRNA and mix thoroughly.
7. Heat the sample in a preheated heat block for 5 min at 90 °C to denature the aRNA.
8. Place on ice to cool for 5 min before loading the samples onto the microarrays. Ensure that no more than 30 min elapses between completion of the denaturation step and loading all of the samples onto microarray slides.
9. Simultaneously with the fragmentation and denaturation of the samples, the slides must be prehybridized. Ensure that the prehybridization buffer has been preheated to 55 °C in a water bath.
10. Place the microarray slides in a slide holder.
11. Immerse the slides in the 55 °C prehybridization buffer and incubate on a rotator for 30 min.
12. Rinse the slides with DEPC-treated water by placing the reservoir holding the slides into the sink and carefully flooding the container with water. Be careful not to let the prehybridization buffer dry on the slides. When all of the prehybridization buffer has been replaced with water, place the reservoir on a rotator and rock gently for 1 min.
13. Dump the water off of the slides and carefully flood the reservoir holding the slides with DEPC-treated water again.
14. Rock gently for 1 min.
15. Repeat the wash and gentle rocking steps three more times, 1 min each wash, for a total of five washes of the microarray slides.
16. Place the slides in a slide holder designed to fit a 96-well plate rotor and dry the slides by centrifugation at 1000 × *g* for 3 min.
17. Adhere a flexible adhesive coverslip to each microarray slide. Line up the end and edges of the adhesive gasket with the end and edges of the microarray and attach the coverslip. Ensure that the gasket is completely adhered at all edges.
18. Thoroughly vortex the first fragmented and denatured sample.
19. Carefully pipet 200 μL of the hybridization sample onto the microarray, loading through one of the ports of the adhered coverslip. Avoid formation of bubbles between the coverslip and slide.
20. Load the remaining samples in the same manner.
21. Use the seals supplied with the adhesive coverslips to seal the ports.

22. Place the microarrays in a slide holder and incubate in a shaking incubator set at 300 rpm or in a hybridization oven at 42 °C for 18 h.
23. Warm a container of Posthybridization buffer #1 large enough to submerge the microarray slides in their slide holder in a water bath set at 42 °C so that it comes to temperature during the hybridization reaction. The first wash step after the hybridization will use this preheated buffer.

### 3.8 Post-hybridization Processing

1. Fill an open container with Posthybridization wash buffer #1. Place a slide holder in a second container and fill with Posthybridization wash buffer #1 to a level that will submerge the microarray slides.
2. Take the microarray slides out of the incubator/hybridization oven.
3. Place the first slide in the open container of buffer and slowly peel off the flexible coverslip such that the surface of the slide is immediately flooded with the buffer.
4. Move the first slide into the slide rack in the second container of buffer, submerged in the buffer.
5. Repeat the removal of coverslips and placement in the slide rack until all microarrays have been processed through this step.
6. To eliminate unbound/unhybridized aRNA from the microarrays, transfer the rack of slides into the container of 42 °C Posthybridization wash buffer #1 (**step 23** in Subheading 3.7). Incubate the microarrays at 42 °C for 1 h exactly.
7. During the incubation, prepare a container of room temperature Streptavidin-Alexa Fluor 647.
8. Remove the rack of slides from the 42 °C Posthybridization wash buffer #1 and submerge in the Streptavidin-Alexa Fluor 647 and incubate for 30 min in the dark (*see* **Note 24**).
9. Set up four containers with Posthybridization wash buffer #2.
10. Transfer the microarray slide rack from the container of fluor into the first container of buffer. Incubate covered for 5 min.
11. Gently raise and lower the slide rack several times in the buffer to gently agitate, then transfer the rack of microarrays into the second container of Posthybridization wash buffer #2. Agitate gently at the beginning and end of the 5 min incubation period.
12. Repeat washing of the slides in the third and fourth containers of Posthybridization washer buffer #2 with gentle agitation at the beginning and end of each 5 min incubation.
13. Prepare a container of Posthybridization wash buffer #3.

14. Transfer the rack of microarray slides to the container of Post-hybridization wash buffer #3.
15. Incubate for 30 s. Maintain gentle agitation throughout the entire 30 s incubation.
16. Dry the slides by centrifugation using a 96-well plate rotor adaptor at 1000 × *g* for 3 min.
17. Store the dried slides in the dark in an opaque slide box.
18. Scan the slides as soon as possible after completion of slide processing, using a GenePix Pro 4000B scanner or equivalent. The scanner and software should be turned on for approximately 15 min in preparation to scan the slides.
19. If the initial scans of the slides identify regions containing high fluorescent background, repeat the final wash step in Posthybridization wash buffer #3, dry the slides by centrifugation as before, and repeat the scan.
20. The microarray data should be analyzed using specialized software such as GeneSpring or a similar program.
21. Use complementary technologies such as RT-qPCR [19] or in situ hybridization [20] to confirm differential gene expression. Similarly, use immunoblot [21], immunohistochemistry [22], or enzyme-linked immunosorbent assay (ELISA) [23] to further explore differential expression of the proteins encoded by differentially expressed genes.
22. The experimental details and raw data from the microarray scans for each microarray experiment should be deposited in a public database at the time of publication to maintain compliance with the recommendations of Minimum Information About A Microarray (MIAME) [24]. The Gene Expression Omnibus (GEO), maintained by the National center for Biotechnology Information (www.ncbi.nlm.nih.gov), is a database designed to store microarray and related data for public use in the USA.

## 4 Notes

1. Make prehybridization buffer in advance and filter; this buffer filters slowly so it is difficult to make in a hurry. Freeze 50 mL aliquots as this volume thaws expeditiously.
2. Cells should readily dissolve in Tri reagent. Pelleted cells that were grown in suspension may need to be dispersed in the Tri reagent to achieve dissolution of the cells. Some especially adherent cells may need to be scraped from the culture dish.
3. If cells grown in different culture dishes are to be combined, add 1 mL of Tri reagent to the first dish. When the cells in that

dish are in solution, aspirate the Tri reagent/cell solution and transfer it to the second dish and continue until all cells to be combined have been dissolved in 1 mL of Tri reagent.

4. Pieces of dissected tissue should be small (e.g., 5 mm in one dimension) as the ability of RNAlater to preserve the RNA in the sample can be overcome if the tissue sample is too large for the volume of reagent (e.g., use five volumes of reagent per mg of tissue).
5. If tissues are from animals, the protocol for animal treatment and tissue procurement must have prior approval from the appropriate institutional committee for ethical treatment of animals, such as the Institutional Animal Care and Use Committee. If tissues are sourced from human subjects, all protocols should conform to the Declaration of Helsinki and must have prior approval from the appropriate institutional committee charged with overseeing the ethical treatment of human subjects in medical research, such as the Institutional Review Board.
6. Use RNase Zap to decontaminate the Polytron probe each time before using it to homogenize tissue. Clean a 100 mL graduated cylinder with RNase Zap and rinse with DEPC-treated water, then fill the cylinder with DEPC-treated water. Spray the homogenizer probe with RNase Zap and rinse with DEPC-treated water, then run the probe in the graduated cylinder of DEPC-treated water to ensure removal of all RNase Zap. Dry off any residual water before using the probe.
7. The 12 × 75 mm tube is well sized for the 7 mm probe. Ensure compatibility of the probe and tube if substituting a different tube or probe.
8. If your tissue samples are especially small or hard to obtain, substitute a 2 mL microfuge tube (Make sure to use a 2 mL microfuge tube; 1.5 mL tubes are too small for the 7 mm probe, and the v-bottoms are typically of a conformation incompatible with the probe.) for the 12 × 75 mm test tube, and use 600 μL Tri reagent instead of 1 mL for homogenization. Homogenize the tissue in this smaller volume of Tri reagent, then rinse the probe in 400 μL of fresh Tri reagent to recover the drop of Tri reagent that would otherwise be lost to the probe. Combine the 600 and 400 μL aliquots. Recovery of this drop results in a significant increase in the total RNA extracted from small tissue samples.
9. Increase the number of pulses of homogenization as necessary for tissues with a greater percentage of connective tissue or muscle. The minced tissue can be soaked in Tri reagent for an hour before initiating homogenization in order to soften especially difficult tissues.

10. The RLT buffer that comes in the RNeasy kit is a lysis buffer and can be used for homogenization of either cultured cells or intact tissues. However, we find that phase separation of RNA from DNA and protein, as occurs with a Tri reagent-based extraction protocol, yields a better product as it provides an additional step that reduces the quantity of genomic DNA that goes onto the column. The greater the quantity of genomic DNA that goes onto the column, the more likely it is that some of that DNA will persist to contaminate the final eluted RNA.
11. Centrifugation separates the sample into a clear, aqueous layer on top containing RNA and an organic layer on the bottom containing protein. Genomic DNA is found at the interface between the two layers. Leave plenty of headspace between the pipette tip and the genomic DNA when pipetting off the aqueous layer containing RNA as it is easier to avoid contamination with genomic DNA than to remove it later. The sample protein can be extracted from the organic layer of the Tri reagent extraction [25]. However, our experience is that some proteins do not perform well in applications such as immunoblot after being denatured in Tri reagent.
12. The volume of RNA is typically approximately 450 μL. Add proportionally more RLT buffer and 100% ethanol if the volume of RNA is greater than 450 μL.
13. Do not use DEPC-treated water for this step as DEPC may interfere with downstream applications of RNA.
14. Recovery of purified RNA is best if the elution is carried out twice with 50 μL water each time than if elution is performed with smaller volumes or only once. If the RNA is too dilute for downstream applications it can be concentrated using a vacuum concentrating device such as a SpeedVac. We have observed that some silica-based columns are not as effective as others at purifying RNA and may compromise the quality and quantity of the eluted RNA. However, we do find that the silica-based columns provide a more consistent product than the phenol–chloroform–isoamyl alcohol RNA extraction method [25].
15. If the RNA concentration is expected to be low, use the Agilent RNA 6000 Pico Chip.
16. Agarose gel electrophoresis and spectrophotometry can be used to analyze RNA [25]. However, the advantage of the Agilent Bioanalyzer is that it consumes only 1 μL of sample to analyze both the quality and quantity of RNA in the sample. This analysis designates the quality of the RNA by an RNA integrity number (RIN). RIN numbers range from a low of undefined to a high of 10; RNA samples for DNA microarray should have RIN numbers greater than 8.

17. Clean the electrodes of the Agilent Bioanalyzer before and after each use, carefully following the manufacturer's instructions. Carry out monthly maintenance of the electrodes and maintain hygiene of the machine to keep it clean and functional.
18. If less than the requisite quantity of 0.2 μg of total RNA of a given sample is available, then the sample can be amplified through two rounds of aRNA. Synthesize aRNA as described in Subheading 3.6 but in the final synthesis step use unlabeled nucleotides since cDNA cannot be synthesized from biotinylated aRNA. Use the unlabeled, single-stranded aRNA from this reaction as a template and use random primers to initiate synthesis of a second round of first strand cDNA. Use T7 oligo dT to prime the synthesis of the second strand of cDNA, and use the resulting double-stranded cDNA to synthesize aRNA. In the synthesis of this second round of aRNA, incorporate biotinylated 11-UTP to yield biotinylated aRNA.
19. When the original total RNA is reverse transcribed to the first strand of cDNA, the reaction uses a primer that carries the T7 promoter sequence plus oligo (dT). The messenger RNA in the sample binds to the oligo dT sequence of the primer. As each new cDNA transcript is synthesized, the T7 promoter sequence is incorporated. In the IVT reaction, the T7 RNA polymerase enzyme in the master mix will transcribe the double-stranded cDNA sequences because they carry the T7 promoter in their 5′ end.
20. Make sure to have the water heated to 55 °C so it is ready for the purification step. Since the IVT reaction proceeds for 14 h, it is often carried out overnight, so it is useful to prepare the water and let it heat overnight so that it will be ready when the IVT reaction is halted.
21. If it is necessary to interrupt the process of synthesizing aRNA, there are several points at which the samples can be frozen at −80 °C until the process can be continued later. It is safe to stop and freeze the samples after synthesis of the second strand of cDNA (*see* Subheading 3.6, **step 6**), after the double-stranded cDNA has been purified (*see* Subheading 3.6, **step 15**), or after elution of the purified aRNA (*see* Subheading 3.6, **step 27**).
22. Do not use DEPC-treated water to dilute aRNA for spectrophotometry as DEPC interferes with the spectrophotometry reading. Instead, use nuclease-free water or purified water with 18.2 MΩ resistivity.
23. Fragmentation of the biotinylated aRNA is designed to lessen secondary and tertiary RNA structure. Secondary and tertiary structure can impede RNA hybridization to DNA microarrays,

hence the biotinylated aRNA is fragmented by metal-induced hydrolysis prior to hybridization.

24. The slides should be kept in the dark as much as possible from this point on, but the photobleaching of Alexa 647 proceeds much more slowly than that of many other fluorescent dyes.

## References


1. Eyster KM, Klinkova O, Kennedy V, Hansen KA (2007) Whole genome deoxyribonucleic acid microarray analysis of gene expression in ectopic versus eutopic endometrium. Fertil Steril 88:1505–1533
2. Gresham D (2011) DNA microarray-based mutation discovery and genotyping. Methods Mol Biol 772:179–191
3. Crujeiras AB, Pissios P, Moreno-Navarrete JM, Diaz-Lagares A, Sandoval J, Gomez A, Ricart W, Esteller M, Casanueva FF, Fernandez-Real JM (2018) An epigenetic signature in adipose tissue is linked to nicotinamide N-methyltransferase gene expression. Mol Nutr Food Res. https://doi.org/10.1002/mnfr.201700933
4. Schaaf CP, Wiszniewska J, Beaudet AL (2011) Copy number and SNP arrays in clinical diagnostics. Annu Rev Genomics Hum Genet 12:25–51
5. Jin X, Guan Y, Shen H, Pang Y, Liu L, Jia Q, Meng F, Zhang X (2018) Copy number variation of immune-related genes and their association with iodine in adults with autoimmune thyroid diseases. Int J Endocrinol 2018:1705478. https://doi.org/10.1155/2018/1705478
6. Chen F, Wang S, Wei Y, Wu J, Huang G, Chen J, Shi J, Xia J (2018) Norcantharidin modulates the miR-30a/metadherin/AKT signaling axis to suppress proliferation and metastasis of stromal tumor cells in giant cell tumor of bone. Biomed Pharmacother 103:1092–1100
7. Luo L, Ji LD, Cai JJ, Feng M, Zhou M, Hu SP, Xu J, Zhou WH (2018) Microarray analysis of long noncoding RNAs in female diabetic peripheral neuropathy patients. Cell Physiol Biochem 46:1209–1217
8. Huang CZ, Xu JH, Zhong W, Xia ZS, Wang SY, Cheng D, Li JY, Wu TF, Chen QK, Yu T (2017) Sox9 transcriptionally regulates Wnt signaling in intestinal epithelial stem cells in hypomethylated crypts in the diabetic state. Stem Cell Res Ther 8:60. https://doi.org/10.1186/s13287-017-0507-4
9. Tan J, Sack BK, Oyen D, Zenklusen I, Piccoli L, Barbieri S, Foglierini M, Silacci Fregni C, Marcandalli J, Jongo S, Abdulla S, Perez L, Corradin G, Varani L, Sallusto F, Sim BKL, Hoffman SL, Kappe SHI, Daubenberger C, Wilson IA, Lanzavecchia A (2018) A public antibody lineage that potently inhibits malaria infection through dual binding to the circumsporozoite protein. Nat Med 24:401–407
10. Cui Z, Xie M, Wu Z, Shi Y (2018) Relationship between histone deacetylase 3 (HDAC3) and breast cancer. Med Sci Monit 24:2456–2464
11. Mateen R, Ali MM, Hoare T (2018) A printable hydrogel microarray for drug screening avoids false positives associated with promiscuous aggregating inhibitors. Nat Commun 9:602. https://doi.org/10.1038/s41467-018-02956-z
12. Cao C, Wu H, Vasilatos SN, Chandran U, Qin Y, Wan Y, Oesterreich S, Davidson NE, Huang Y (2018) HDAC5-LSD1 axis regulates antineoplastic effect of natural HDAC inhibitor sulforaphane in human breast cancer cells. Int J Cancer 143(6):1388–1401. https://doi.org/10.1002/ijc.31419
13. Quinones B, Lee BG, Martinsky TJ, Yambao JC, Haje PK, Schena M (2017) Sensitive genotyping of foodborne-associated human noroviruses and hepatitis A virus using an array-based platform. Sensors 17:2157. https://doi.org/10.3390/s17092157
14. Wnuk A, Rzemieniec J, Litwa E, Lason W, Kajta M (2018) Prenatal exposure to benzophenone-3 (BP-3) induces apoptosis, disrupts estrogen receptor expression and alters the epigenetic status of mouse neurons. J Steroid Biochem Mol Biol 182:106–118. https://doi.org/10.1016/j.jsbmb.2018.04.016
15. Piaggi P, Masindova I, Muller YL, Mercader J, Wiessner GB, Chen P, SIGMA Type 2 Diabetes Consortium, Kobes S, Hsueh WC, Mongalo M, Knowler WC, Krakoff J, Hanson RL, Bogardus C, Baier LJ (2017) A genome-wide association study using a custom genotyping array identifies variants in *GPR158*

associated with reduced energy expenditure in American Indians. Diabetes 66:2284–2295

16. Wang R, Lei T, Fu F, Li R, Jing X, Yang X, Liu J, Li D, Liao C (2019) Application of chromosome microarray analysis in patients with unexplained developmental delay/intellectual disability in South China. Pediatr Neonatol 60(1):35–42. https://doi.org/10.1016/j.pedneo.2018.03.006
17. Revay T, Oluwole O, Kroetsch T, King WA (2017) *In vivo* and *in vitro* ageing results in accumulation of *de novo* copy number variations in bulls. Sci Rep 7:1631. https://doi.org/10.1038/s41598-017-01793-2
18. Dai J, Qiu W, Wang N, Nakanishi H, Zuo Y (2018) Comparative transcriptomic analysis of the roots of intercropped peanut and maize reveals novel insights into peanut iron nutrition. Plant Physiol Biochem 127:516–524
19. Booze ML, Eyster KM (2016) The use of real-time reverse transcription-PCR for assessing estrogen receptor and estrogen-responsive gene expression. Methods Mol Biol 1366:19–28
20. Fietz D, Bergmann M, Hartmann K (2016) In situ hybridization of estrogen receptors α and β and GPER in the human testis. Methods Mol Biol 1366:189–205
21. Chakrabarti S, Davidge ST (2016) Analysis of G-protein coupled receptor 30 (GPR30) on endothelial inflammation. Methods Mol Biol 1366:503–516
22. Khan F, Ricks-Santi LJ, Zafar R, Kanaan Y, Naab T (2018) Expression of p27 and c-myc by immunohistochemistry in breast ductal cancers in African American women. Ann Diagn Pathol 34:170–174
23. Kramer B, Kneissle M, Birk R, Rotter N, Aderhold C (2018) Tyrosine kinase inhibition in HPV-related squamous cell carcinoma reveals beneficial expression of cKIT and src. Anticancer Res 38:2723–2731
24. Brazma A, Hingamp P, Quackenbush J, Sherlock G, Spellman P, Stoeckert C, Aach J, Ansorge W, Ball CA, Causton HC, Gaasterland T, Glenisson P, Holstege FC, Kim IF, Markowitz V, Matese JC, Parkinson H, Robinson A, Sarkans U, Schulze-Kremer S, Stewart J, Taylor R, Vilo J, Vingron M (2001) Minimum information about a microarray experiment (MIAME)-toward standards for microarray data. Nat Genet 29:365–371
25. Rodrigo MC, Martin DS, Redetzke RA, Eyster KM (2002) A method for the extraction of high-quality RNA and protein from single small samples of arteries and veins preserved in RNAlater. J Pharmacol Toxicol Methods 27:87–92

# Chapter 3

# Data Warehousing with TargetMine for Omics Data Analysis

Yi-An Chen, Lokesh P. Tripathi, and Kenji Mizuguchi

## Abstract

Most biological processes including diseases are multifactorial and determined by a complex interplay of various genetic and environmental factors. This chapter aims to provide a user guide to data querying, analysis, and visualization with TargetMine and the associated auxiliary toolkit. We have also discussed some of the commonly used data queries for the researchers who are interested in gene set analysis within a data warehouse framework. Overall, TargetMine provides a convenient web browser-based interface that enables the discovery of new hypotheses interactively, by performing analysis of omics data using complicated searches without any scripting and programming efforts on the part of the user and also by providing the results in an easy-to-comprehend output format.

**Key words** Data warehouse, Data integration, Multi-omics data analysis, Drug discovery, Gene prioritization, Data mining, Knowledge discovery

## 1 Introduction

The development of high-throughput "omics" technologies has contributed to a proliferation of genomics, transcriptomics, proteomics, and metabolomics data for a wide range of biological systems, including diseases. However, omics experiments typically generate large volumes of data; for instance, the list of differentially expressed genes (DEGs) generated by a gene expression microarray experiment can easily run into hundreds and even thousands. This data overload far outstrips the pace at which individual genes can be characterized experimentally. Therefore, there is a pressing need for the development of bioinformatics tools and approaches that can process and analyze such large volumes of data for prioritizing candidate genes and for biological knowledge discovery.

A single omics datatype, such as microarray data, on its own typically offers very specific (and limited) insights into gene function in the biological phenomenon under study. Therefore,

Yi-An Chen and Lokesh P. Tripathi contributed equally to this work.

Verónica Bolón-Canedo and Amparo Alonso-Betanzos (eds.), *Microarray Bioinformatics*, Methods in Molecular Biology, vol. 1986, https://doi.org/10.1007/978-1-4939-9442-7_3, 

combining multiple omics data types can complement the information in the individual data types and deepen our understanding of the causative mechanisms underlying the complexity within the cellular processes and complex diseases and disorders. However, integrating diverse biological datatypes into a singular, holistic framework is an extremely challenging task because of the diversity of data types, variability in technical platforms and experimental methodologies. In recognition of the benefits of data integration many different types of frameworks and methods have been developed to integrate and collate diverse data types, with their own associated strengths and shortcomings [1].

A data warehouse is a data integration system that assembles all data onto a unified platform and therefore is well suited for integration of multiple omics data types and to query and analyze diverse data types in a singular framework [2–4]. We have developed TargetMine, an integrated data analysis platform for gene set analysis and biological knowledge discovery [5, 6] based on the multipurpose InterMine data warehouse framework [7]. TargetMine has been successfully employed for analysis of omics data and candidate gene prioritization [8–13].

This chapter is aimed at providing a basic user guide for researchers seeking to use TargetMine for integrative analysis of large gene (and protein) sets emanating from high-throughput omics experiments such as gene expression microarrays. We introduce the TargetMine data analysis platform and discuss in detail how the users can utilize the various embedded tools and features to search, visualize, and analyze the integrated biological data within TargetMine. We discuss how the users can perform keyword searches, submit batch queries, manipulate different gene/protein lists, navigate the TargetMine data model to perform complicated queries, and perform functional enrichment analysis. We also describe how the users can obtain the results in the form of exportable tables that can be immediately inputted to other stand-alone data analytical and visualization tools for subsequent analysis.

## 2 Materials

1. TargetMine and the associated auxiliary tool kit are freely available for all users at http://targetmine.mizuguchilab.org, subject to terms of use. No programming expertise is necessary for the users to perform data analysis with TargetMine; advanced users can, however, use the TargetMine Application programming interface (API), available in commonly used programming languages to access all the TargetMine features and to perform complicated queries and data analysis.

2. Users and developers may also install a local version of TargetMine, the TargetMine source code and instructions describing installation are available at http://targetmine.mizuguchilab.org/download. TargetMine is typically rebuilt once a month and the latest releases are hosted on https://github.com/chenyian-nibio/targetmine. The local installation of TargetMine requires IT expertise and the users are recommended to obtain the necessary technical support from their institutional IT support or a commercial service provider. To install TargetMine locally for commercial purposes, the users should contact our commercial service partner. (http://www.mss.co.jp/businessfield/bioinformatics/solution/products/targetmine/ index.html).

## 3 TargetMine Data Model, Development and Data Sources

1. TargetMine is based on the InterMine data warehouse framework [7] that uses an object-oriented data model; each biological entity (such as genes and proteins) in TargetMine/InterMine is modeled as an "object" that is described by a set of "attributes"; objects of the same type are grouped together in "classes" and the interclass relationships and associations are modeled as "references." The InterMine data structure readily allows the stored biological data to be easily navigable.
2. The TargetMine data model was developed by combining a customized version of the core InterMine data model and models specifically developed for TargetMine to process and analyze a wide variety of omics data types and that would assist in drug discovery and related research. For instance, transferring experimental readings from probes/probesets to genes is an important step in analyzing microarray data. We specifically designed a probeset data model to store the mapped associations between selected Affymetrix probesets and their corresponding genes (as supplied by the manufacturer) and thereby also linking them up with the gene-associated annotations. The implementation of the probeset data model allowed the users to directly query a probeset or a collection of probesets such as a set of differentially expressed probes derived from a microarray experiment and retrieve their functional associations and perform biological enrichment analysis (see below).
3. The data sources in TargetMine were carefully selected to survey a wide expanse of biological data space to facilitate target prioritization and assist biological knowledge and drug discovery in general [5, 6, 14]. As of now the data sources are limited to human, mouse, and rat, the model organisms in disease biology. TargetMine is rebuilt once a month to incorporate the updated data sources as well as to integrate new data

sources that we have identified as potentially useful for drug discovery. The list of current data sources and their version releases may be found at http://targetmine.mizuguchilab.org/targetmine/dataCategories.do.

4. We have also developed the TargetMine Auxiliary Toolkit to assist with data analysis and visualization and to transition TargetMine from a data warehouse to a more integrative data analysis platform.

## 4 Data Querying and Gene Set Analysis

On the home page for the TargetMine data analysis platform, http://targetmine.mizuguchilab.org, the users will find the links for the TargetMine data warehouse and the TargetMine Auxiliary Toolkit (see below). The users can click the "Start using TargetMine" button to commence data analysis with TargetMine (Fig. 1).


**Fig. 1** Getting started with TargetMine. TargetMine homepage provides information that the users are advised to go through before getting started. The tabs on the top provide links to an overview of TargetMine system, tutorials, latest updates, documentation and information on bulk data download. Clicking the "Start using TargetMine" button transfers the users to the TargetMine main page; "TargetMine Auxiliary Toolkit" button provides the link to an overview of the auxiliary toolkit, and clicking the "Enrichment analysis+" and "Composite network" buttons transfers the users to the corresponding analyses pages in the auxiliary toolkit. On the TargetMine main page, the users can enter a keyword in the search box to perform a keyword search or paste list of object identifiers in the list upload box to analyze a list. The tabs on the top of the page provide links to the predefined queries (templates), list upload function, the "Query Builder" tool and the TargetMine API information. The account users can "Log in" and click the "MyMine" tab to manage the stored lists and queries. Source: http:/targetmine.mizuguchilab.org/

### 4.1 Keyword Search

1. The simplest type of query that the users can perform in TargetMine is searching by keywords. In the TargetMine home page, users can enter the keyword (Gene/protein ID/miRNA/chemical compound ID, symbol, name, pathways, ontology terms, etc.) in the "Search" box and then click the "SEARCH" button below. Alternatively, the users can enter the keywords in the "Search:" box on the upper right of any TargetMine page and click "GO" (Fig. 2).
2. The keyword search retrieves a list of items that ranked by the similarity to the query keywords; on the left side, the hits are categorized by item type (genes protein, terms, etc.) and organism along with the number of hits of each type that were returned in the search results. The user can click the results to view detailed information about the selected items (Fig. 2).
3. Clicking on the selected result will direct the user to the type-specific report page that provides an overview of the annotations associated with selected result that are divided into different sections such as different types of biological theme associations (Gene Ontology [GO] [15], KEGG/Reactome/NCI-PID Pathways [16–18], etc.), biomolecular interactions (table-wise and as embedded graphics) and orthologous associations if available. The user can select a specific section from the "Quick Links:" tab below the summary or by scrolling down. The user can also follow the external links on the right of the report page to examine more detailed information associated with the selected object (Fig. 2).

### 4.2 Uploading and Manipulating Data Lists in TargetMine

1. The "Upload" function under the "Lists" tab allows the user to upload a list of items (such as genes or proteins) to TargetMine. TargetMine accepts different types of identifiers; for instance, a gene list could consist of NCBI Gene IDs, gene symbols, ENSEMBL IDs, RefSeq or GenBank IDs. The users may either type/paste the list of identifiers into the box or use the "Browse..." button to upload identifiers from a file. The users must specify the data type (genes, proteins, probesets, etc.) of the list items from the drop-down menu next to the "Select type:" field. When uploading identifiers that can be less than specific, such as gene symbols or RefSeq IDs, the users may also click the checkboxes next to "Match on case" and "Remove version tags" to minimize ambiguity. Finally, the users can create a list by clicking "Create List" button at the bottom of the menu (Fig. 3).
2. Once the list is uploaded, the user is asked to "Choose a name for the list" in the field provided and also to verify if the identifiers have been mapped correctly. The centralized data model in TargetMine enables the user-supplied IDs to be

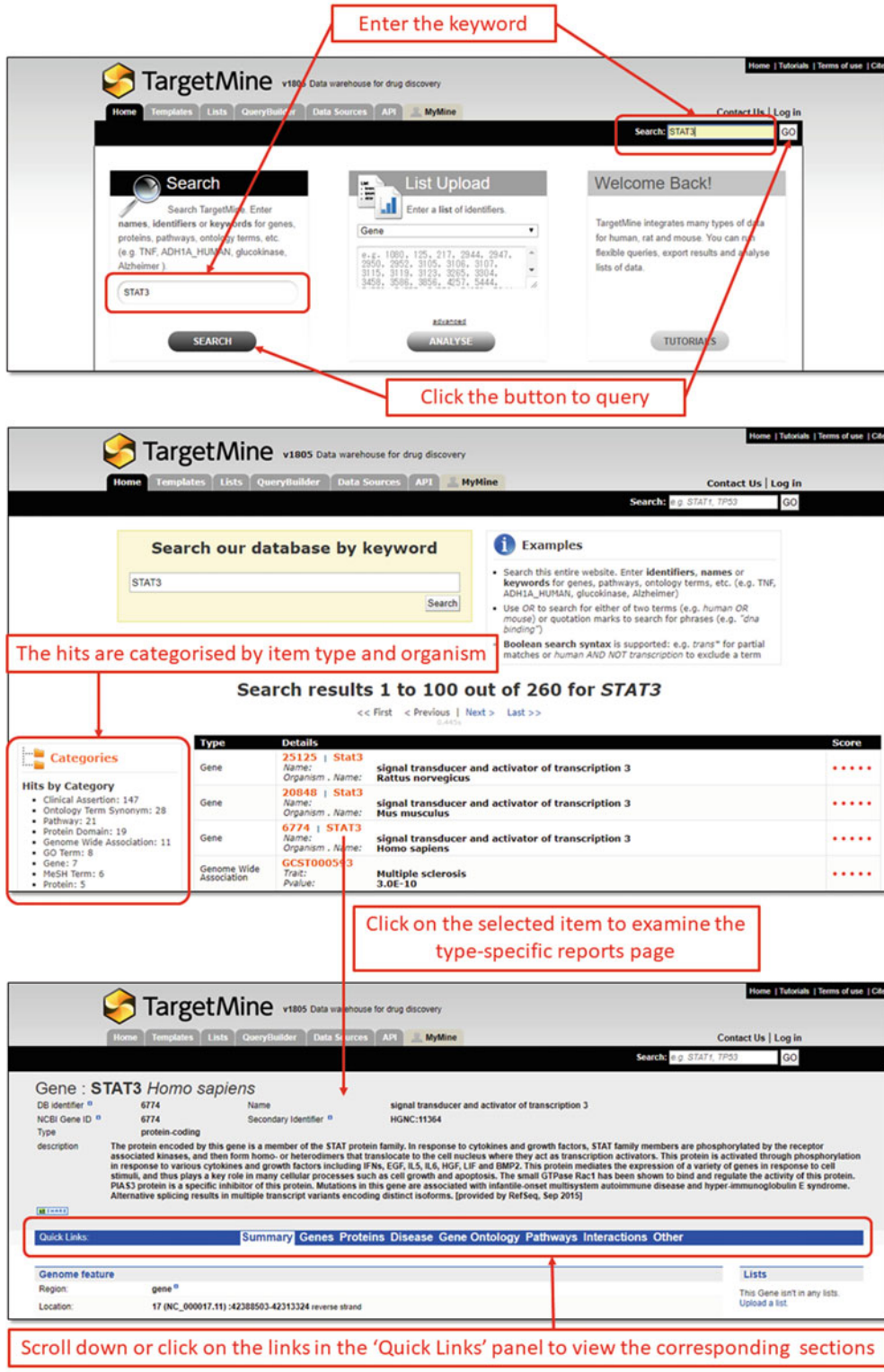


**Fig. 2** Keyword search in TargetMine. A search for the keyword "STAT3" retrieved 260 results containing the search term; the results were listed in the order of relevance that was highlighted as a score on the right of the individual results. The results were also categorized by item type and organism under "Categories." Clicking on a selected item transfers the user to the object reports page that provides a summary of the object being examined

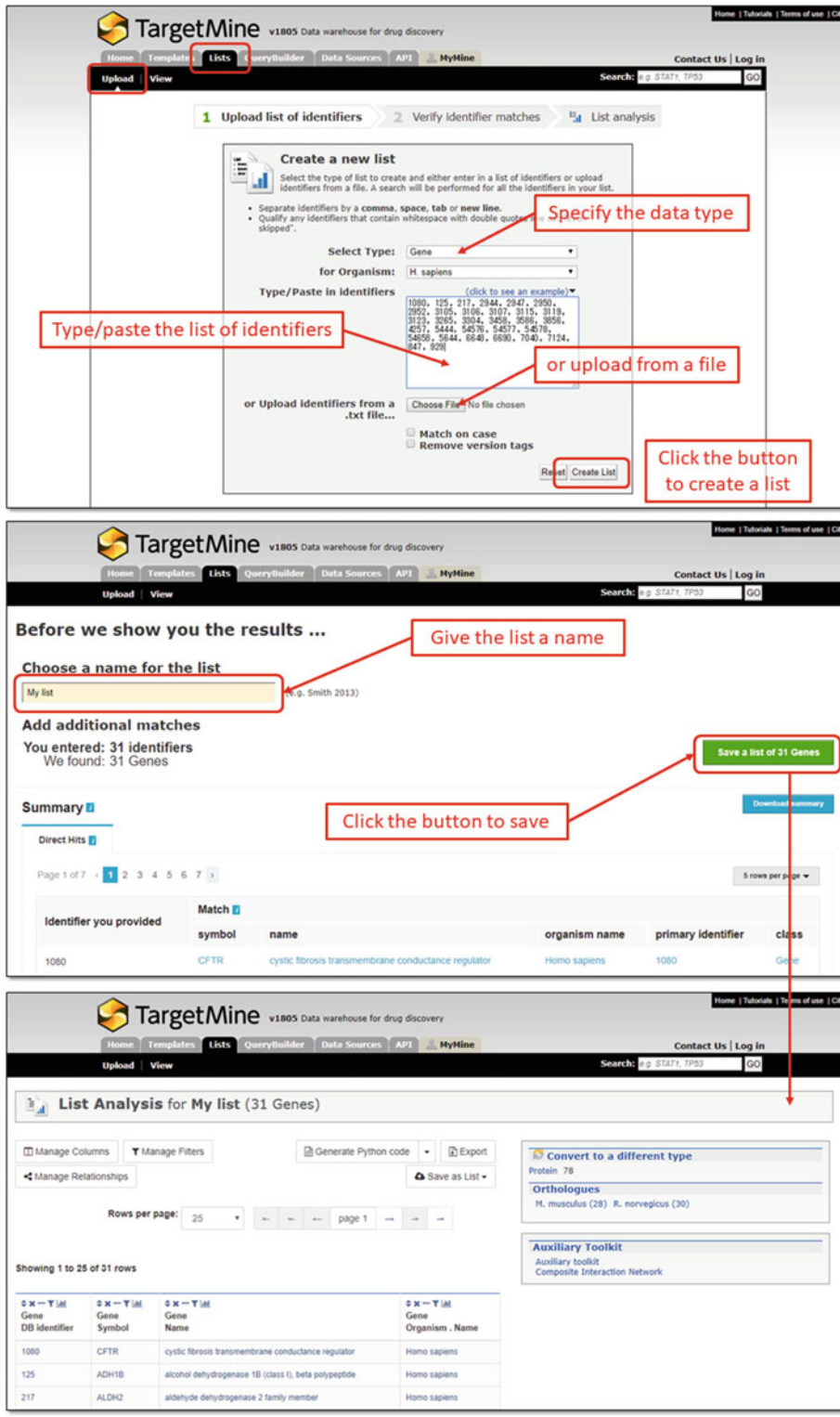


**Fig. 3** Uploading and creating a gene list. The list upload function can be accessed from the "Lists" tab in every TargetMine web page. The user is required to specify the object type and either type/paste common

accurately mapped to the database entries in TargetMine. In the event of multiple mappings, the users have the option to select some or all of the mappings or ignore them altogether. The deprecated and unrecognized identifiers will be displayed in the list of objects not found and excluded from the final list. Finally, the users are asked to save the list by clicking the "Save a list of # <>", where # <> refers to the number and type of mapped entities in the user-supplied list (Fig. 3).

3. Next, the user is directed to the TargetMine "List Analysis" page that provides a table of objects in the user-supplied list and accompanying information. For instance, for a user-supplied gene list, the page provides NCBI gene IDs, official gene symbols, gene names, and the source organism. The table can be exported to external files in excel and text compatible formats or upload to the Galaxy tool [19] by clicking on the "Export" button and select from the available options. The "Manage Columns" button enables the users to select the columns to be retained or deleted in the exported list or to select and include additional columns in the list (see below) (Fig. 3).
4. The users can retrieve their lists by clicking on the "View" tab under the "Lists" tab. The user uploaded lists will be retained while the web browser session is active. For long-term storage of lists, the users are advised to create a TargetMine user account, which will allow them to store and categorize their lists for future reference. The stored lists can be accessed and also shared with other account users by using the personalized "MyMine" tab (Fig. 4).
5. The "Lists" tab also allows the users to perform multiple actions with two or more lists containing objects of the same type. The user needs to click the checkboxes next to the lists that they wish to manipulate and process and then select the operation they wish to perform from the available "Actions"—"Union," "Subtract," "Intersect," and "Asymmetric Difference"—on the top of the panel. Once the user has selected the lists and the appropriate "Action," they will be prompted to enter a name for the resultant list and click "Save" to generate a new TargetMine list. It should be noted that the "Asymmetric Difference" operation can only be performed with two lists at a time and when performing this action, the users will be prompted to enter the names of two resultant lists—List A

**Fig. 3** (continued) identifiers in the corresponding box or browse to upload a list from a local file and click the "Create List" button. On the next page, the user is required to enter a name for the list and click the "Save. . ." button to create the list. The user is then transferred to the "Lists Analysis" page to view the objects in the list and perform further actions—functional enrichment, template queries, etc.

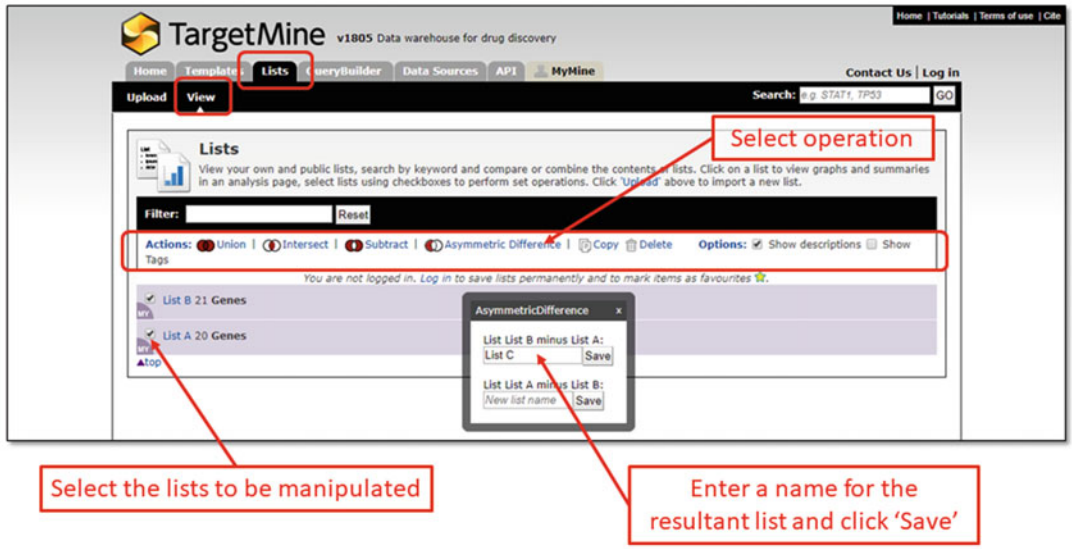


**Fig. 4** List operations in TargetMine. The "View" tab allows the user to retrieve all the created lists. The "Actions" panel features the different types of operations that can be performed to manipulate two or more lists containing the same object type. In this example we have performed the operation to obtain an "Asymmetric difference" of "List A" and "List B" containing 20 and 21 genes, respectively. After selecting the checkboxes next to the two lists and clicking "Asymmetric Difference" in the "Actions:" panel, the user is prompted to enter a name for the resulting lists (in this example, List C, which is a list of objects derived after subtracting the contests of List A from List B) and clicking the "Save" button to create a new list

"minus" List B and List B "minus" List A. The users can choose to name and save either or both of the resultant lists (Fig. 4).

### *4.3 Advanced Data Mining with TargetMine*

1. The TargetMine "QueryBuilder" is a powerful and a versatile tool that allows the user to (a) easily navigate the TargetMine data model across multiple data types, and (b) choose and constrain from the available biological associations to be attached to the input set of items and obtain a compact and unified output for subsequent analysis.
2. The "QueryBuilder" tool can be accessed from the tab panel in the top of every web page in TargetMine. To use the "QueryBuilder," the first step is to select the query type from the list of entities in the drop-down menu and click select. This action transfers the user to the QueryBuilder page that outlines the construction of the user-defined query. The query builder consists of three sections that display the data model, query outline, and output columns. The top left panel displays the "Model browser" where the user can select the data types and the associated information to be displayed and/or constrained for the query. To view and select the type of information available within the "Model browser," the users can click the

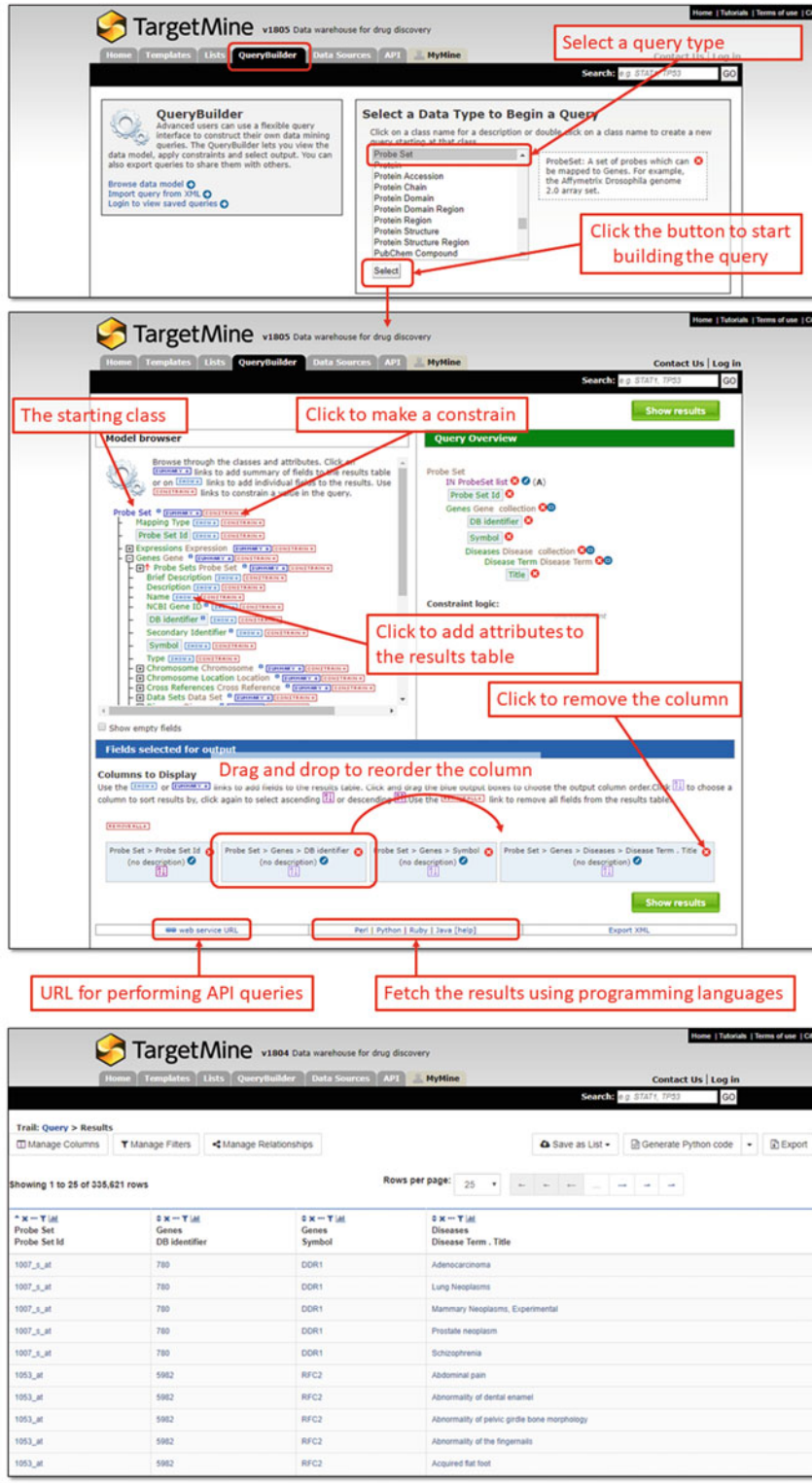


**Fig. 5** Navigating the TargetMine data model using the "Query Builder." The "Query Builder" tool requires the users to specify the query data type. The "Model browser" displays the class of the user-selected data type;

"+" symbol to expand the class/node of interest and view the available nodes/classes within; the "SHOW" label allows the users to select the datatypes and attributes to be displayed in the output, and the "CONSTRAIN" label allows the users to restrict the queries to user-specified subsets of the initial dataset (such as the user-supplied gene set and/or specific attributes). The top right panel displays the "Query Overview" that summarizes the datatypes that were selected and constrained for query using the "Model browser." The "Model browser" is equipped with two features that allow the users to either remove an existing constraint by clicking on the red "×" icon, or modify the initial constraints by clicking the blue "edit" icon. Lastly, the "Fields selected for output" panel at the bottom shows the arrangement of the "Columns to Display" that is the user-selected attributes as they will appear in the results table. The users can rearrange the column order by clicking and dragging the column of interest to the desired position. The user can finally execute the query and view the output by clicking on the "Show results" button in the right of the panel. Advanced users can click the "web service URL" to retrieve the URL for performing API queries; the web service client libraries to perform API queries in TargetMine and fetch the results in different formats are available in Perl, Python, Ruby, and Java programming languages. The users can also export the query in xml format by clicking on the "Export XML" button (Fig. 5).

3. *Templates* To enable the users to quickly and easily perform different queries across diverse data types, TargetMine is equipped with a library of "Templates" that consist of predefined queries with a simple form and description. TargetMine Templates are categorized by data types, thus allowing the users to search for the desired query by a keyword in the field provided next to "Filter" or to filter them by selecting the data category from the drop-down menu. Templates can be easily modified by imposing constraints (filters) to restrict the query and output to user-specified subsets of data. In a template query, the user may directly input a string (a gene name or gene ID for instance) in the field provided next to the "LOOKUP" function (the user may enter multiple gene

**Fig. 5** (continued) the user can click the "SHOW" button next to individual attributes to select the information to be displayed in the output and/or click the "CONSTRAIN" button to specify the constraint. Clicking the "SUMMARY" button displays a set of default attributes for the data type. The user-selected attributes and constraints are displayed in the "Query Overview" panel on the right. The attributes selected for display in the final results are added to "Fields selected for output" panel at the bottom of the query builder. The user can run the query by clicking the "Show results" button and generate a results table

names or IDs by separating them individually by a comma) or constrain the query to the user-supplied list using the "constrain to be" feature and selecting from the premade TargetMine lists from the drop-down menu next to "saved Gene list". The user can finally execute the query and view the output by clicking on the "Show results" button on the lower-left end of the template form. The user can also customize the template query by clicking the "Edit Query" button on the lower right; the user will be directed to the query builder web page to perform the requisite edits. A commonly used template search in TargetMine is "Gene(s) → HCDP (tissue constrained)," that is, given a gene or a list of genes, retrieve the high confidence direct physical protein–protein interactions (PPIs) (HCDP; see below), where the interacting genes are highly expressed in user-specified tissues and cell types (Fig. 6). This is an example of a template that queries for functional associations across three different TargetMine classes—"Genes," "Interactions," and "Expression" that describe genes, PPIs, and organ/cell-specific gene expression, respectively. The account users can also store their preferred templates under the "MyMine" tab for reuse or for sharing with other users.

4. *Results tables* As with the lists, TargetMine query outputs are displayed as a table of results. The table appears with the columns in the order of display as specified by the user in the query builder or predefined in the templates. The users can specify the number of rows to be displayed in a single web page by selecting from the drop-down menu next to "Rows per page"; the default view is 25 rows per page. TargetMine tables enable the users to examine, analyze, and manipulate the results—either column-wise or the table as a whole. Every column is equipped with set of features (visible as icons at the top of the column) to perform column operations. The users can sort the column in ascending or descending using the triangle-shaped icon; the upward facing triangle "▲" indicates that the column is sorted in an ascending order, whereas the inverted triangle "▼" indicates that the column is in a descending order. The users can also remove the column using the "×" icon; toggle the column visibility using the ellipsis ("...") icon, the column can be reopened for viewing using the expand icon "↔"; filter the contents of the column using the funnel-shaped icon and view the summary of the column objects using the bar chart icon. The view column summary function also allows the users to filter and display selective column entries by selecting the checkboxes next to the entries in the display and clicking on the "Filter" button at the bottom of the display. Alternatively, the user can perform an inverse select and filter operation by selecting the desired checkboxes and clicking on the "Toggle selection" icon. The "Reset selection" icon allows


**Fig. 6** An example of TargetMine template query. In this example we demonstrate the "Gene(s) → HCDP (tissue constrained)" template to find all the high-quality binary PPIs (HCDPs) that are associated with the gene of interest and are highly expressed in specific cells/tissues. Users have the option to enter the query gene (or select a preuploaded list of genes) and specifying the cell or tissue to filter the interacting genes

the users to undo the previous operations in the display. An "undo" icon above the column allows the user to undo their last operations. Furthermore, the user can manage the columns by clicking the "Manage Columns" button; clicking the button will display a pop-up window, where the user can reorder the columns by dragging; remove columns either by clicking on

the red (−) icon to the right of the column names or dragging the column to the trash icon; sort the column order using "Sort order" button; and add additional columns by clicking on the "+ Add a Column" button and choose from the available attributes in the model browser (Fig. 7). The user can also

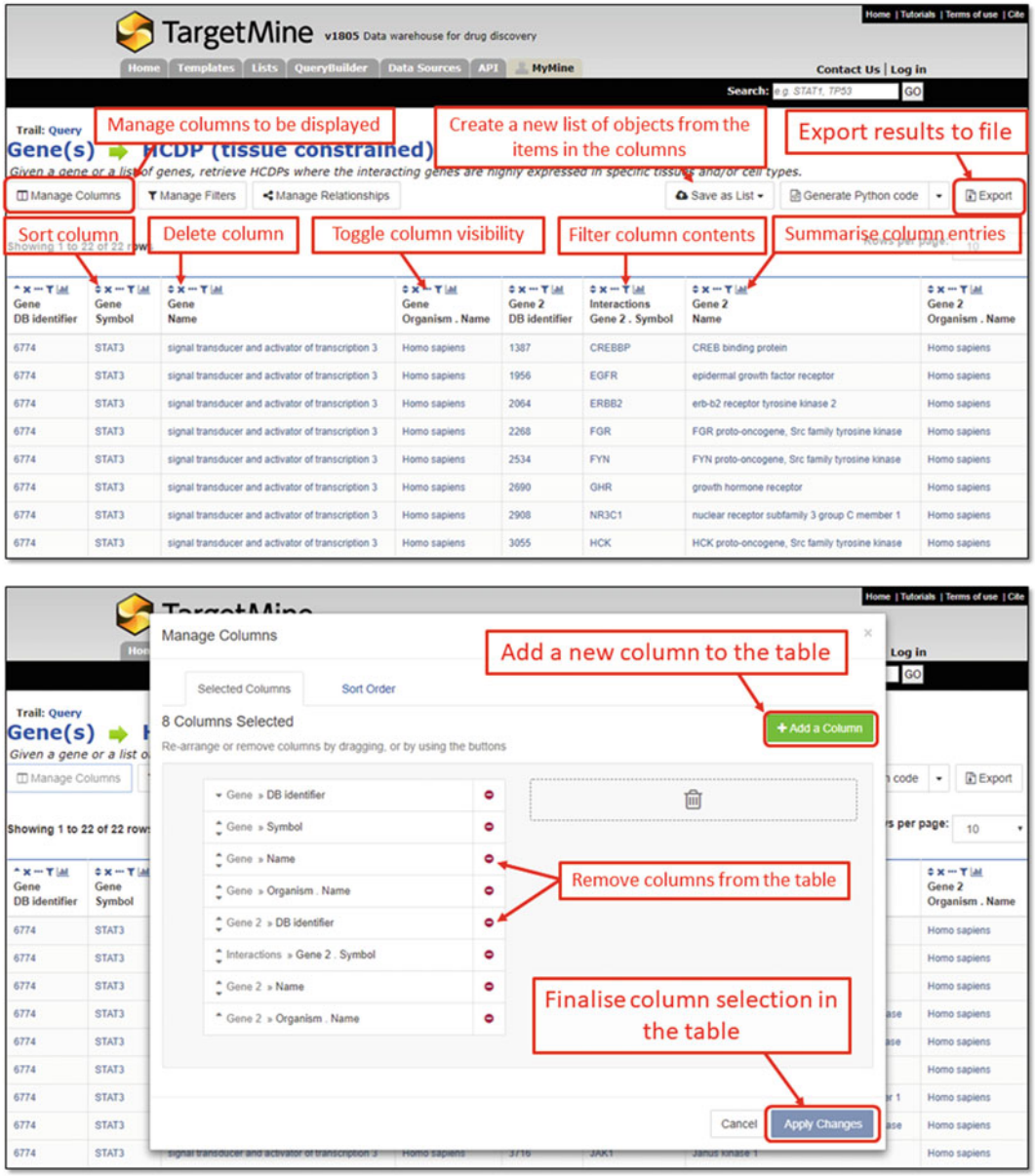


**Fig. 7** TargetMine tables display query output results or information for a list of objects. Users can add and remove columns, customize existing filters, and manage the attribute relationships for column objects using the features on the upper left of the page. Users can save the contents of individual columns as lists and export the table in different formats. Each column header has multiple icons that allow the user to sort, filter, and summarize column contents and manipulate column visibility

add new filters or modify existing filters using the "Manage Filters" function. Moreover, the users can determine how the different attributes are presented and stored in the table using the "Manage Relationships" function. The "Save as List" function allows the user to generate a new list of objects from the objects listed in the table. The user can choose to create a new list of objects by clicking on the "Create List" button and enter the name of the list when prompted; the users can also append the selected objects to an existing list (of the same type) by using "Add to List" function. The users can further click the "Pick items from the table" and select the checkboxes next to the individual objects to select the objects to be stored in the resulting list. Finally, the "Export" function allows the users to export the table in multiple formats and also specify the columns and rows to be included in the exported file. The users can preview the output by clicking on the "Preview" tab (limited to three results) and download the file by clicking the "Download" button.

### 4.4 Biological Enrichment Analysis in TargetMine

1. Microarray and similar omics experiments can often yield hundreds or even thousands of genes that need to be properly analyzed to understand the underlying biological process. Functional enrichment analysis is based on the statistical analysis of the relative abundance of biological themes (including, but not limited to, KEGG/Reactome/NCI-PID pathways [16–18], Gene Ontology (GO) terms [15], and Integrated Pathway clusters (IPCs) [14]) and allows the users to identify the themes (and the associated genes/entities) that are over-represented, that is, enriched and therefore most pertinent to the biological conditions under study.
2. Enrichment of specific biological themes associated with the user-supplied list of gene/entities seeks to identify genes/entities and the associated biological themes that are more abundant than that would be expected to occur by chance given a background population (by default, the number of genes/entities mapped to the specified biological theme in the given genome); the *p*-value of the degree of the abundance is estimated by one-tailed Fisher's exact test. Since multiple statistical tests when performed in parallel can increase the probability of false positives, the inferred *p*-values are further adjusted for multiple test corrections to control the false discovery rate. The users can set their own *p*-value thresholds by selecting from the drop-down menu under "Max p-value." The users may also click the drop-down menu under "Test Correction" to choose from the available correction methods for multiple testing—Benjamini–Hochberg [20, 21], Bonferroni [22], and

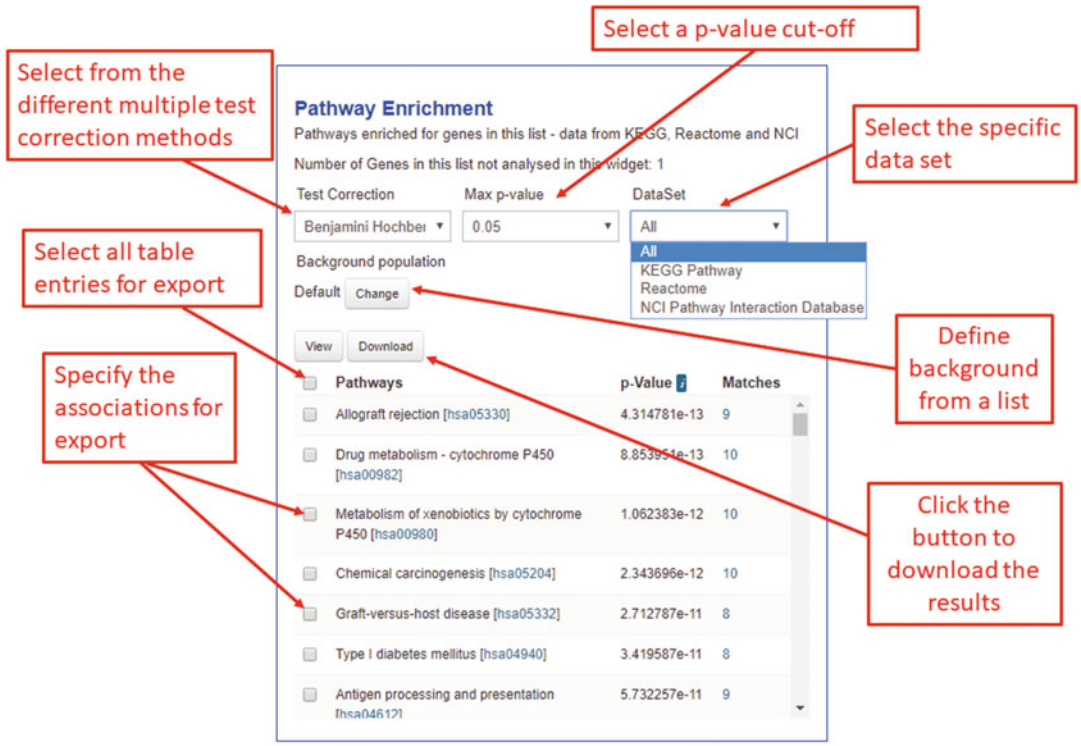


**Fig. 8** Enrichment widgets output the results of statistical tests for the enrichment of specific biological themes (pathways, GO terms, etc.) in the user-created lists in a tabular form. In this example, the users can specify the pathway dataset from the KEGG, Reactome, and NCI-PID repositories or choose all of them, select *p*-value thresholds and multiple-test correction options. The users can also select a background population for the enrichment analysis. The users can export the results by clicking the "View" or "Download" buttons

Holm-Bonferroni [23]—that offer varying degrees of stringency when filtering out false positives (Fig. 8).

(Note: The statistical calculations are performed for a single species only).

3. Enrichment statistics for different biological themes are summarized in the individual enrichment widgets. The users can also perform the enrichment analysis with a customized background population (the number of genes/entities tested in the underlying experiment for instance) for an individual widget by clicking on the "Change" button under "Background population" and choose from the available lists (that were uploaded by the user prior to the selection). The background population so selected must, however, include all the items contained in the list and must be selected individually for each of the enrichment widgets (Fig. 8).
4. The individual enrichment widgets display the widget-specific enriched biological themes along with their associated *p*-values

and the corresponding "Matches," that is, the counts of the objects in the query list that were associated with the given biological theme. By default, only the significantly enriched biological associations, that is, those that satisfied a condition of $p \leq 0.05$ after a multiple test correction with the Benjamini and Hochberg procedure [20, 21] are displayed in the table (Fig. 8).

5. Enrichment widgets are designed to output the results of multiple enrichment analyses for biological themes of the same data type, where available. For instance, the "Pathway Enrichment" widget allows the users to individually examine the enrichment of KEGG Pathways, Reactome pathways, NCI-PID pathways, or collectively for all of them by selecting the corresponding dataset from the drop-down box below "Data Set" (Fig. 8). Likewise, the users can specify the preferred GO aspect (biological process, cellular component, and molecular function) in both the "GO Enrichment" and "GOSlim Enrichment" widgets. Also, the users may specify the background dataset for the protein–chemical compound (small molecule) interactions (PCIs) in the "Compound Enrichment" widget.
6. The users can export the enrichment results by clicking on the checkbox next to the <Theme type> and then either clicking the "View" button to export the results into a TargetMine table or the "Download" button to export the results in text or excel compatible formats. The users can specify select individual associations by clicking on the corresponding checkboxes in the enrichment widget; doing so will only export user-specified enriched associations upon clicking the "View" or "Download" buttons (Fig. 8).

## 5 Network Analysis with Multiple Biomolecular Interaction Types

1. Most genes and proteins function in concert with other biomolecules such as proteins, nucleic acids, and metabolites. Typically, such interactions are represented as a network model which consists of individual biomolecules as *nodes* and biomolecular interactions as *edges*, that is, the vertices connecting them. Protein–protein interactions (PPIs) are key determinants of much of the cellular function and therefore prominent candidates for knowledge and target discovery. In this section, therefore, we will discuss network analysis largely in the context of PPI networks (PPINs).
2. TargetMine users can query for PPIs for their initial list of genes/proteins to construct context-specific PPINs that can help to obtain a deeper understanding of gene functions via

*PPIN analysis* (see below). For convenience, all PPIs in TargetMine are stored as gene–gene interactions.

3. In the "Model browser," the users can retrieve PPIs for a gene or a list of genes by clicking on the "SUMMARY" label associated with the "Interactions" class. Since, the publicly available (unfiltered) PPI data can be inherently noisy, to ensure the robustness of their PPIN analysis, users can choose to restrict their searches to a subset of PPIs by expanding the "Interactions" class and the "Confidences" class within and then clicking the "CONSTRAIN" label associated with "Type"; the user will then be prompted to select a filter specifying the prefiltered PPI type from the drop-down menu to "Add to the query." Two types of prefiltered PPI subsets are available within TargetMine—"HC" (high-confidence PPIs) are PPIs characterized by at least two different experimental methods or reported in two independent publications and "HCDP" (high-confidence direct physical PPIs) that are a subset of HC that were classified as binary PPIs and are most suitable for PPIN analysis [6].
4. *PPIN analysis* In the first step toward construction of PPINs, PPIs (typically HCDPs) for the genes/proteins within the user-supplied list are retrieved as described above. The resulting PPIs can be stored as a separate list; the user may choose to create an extended gene list by a union of the initial gene list and the list of interacting genes/proteins. Alternatively, the user can append the interacting genes to the initial list (as described in column operations above) to create an extended gene list. The former option is useful if the user wishes to reuse the initial gene list for other tasks. The extended gene list represents the components of the inferred PPIN involving user-supplied genes and the interacting genes. To construct and visualize the actual PPIN, the user must perform an additional query to recreate all the HCDPs between all the genes in the extended list. This operation can be achieved using the query builder or more simply by using the template for "Gene(s) → Intraset HCDP" that is given a list of genes, retrieve all HCDPs within the list to construct a PPIN for the user-supplied gene list. The PPIN can be exported using the column functions described above and the network components can be further examined and processed either using stand-alone graphical tools such as Cytoscape [24] or using the Composite interaction network feature in the TargetMine Auxiliary Toolkit [6] (see below).
5. *Network topology* The connectivity of nodes and edges within a PPIN is referred to as *network topology. Network topology* is a key determinant of network function, therefore network concepts such as *node degree distribution* and *betweenness centrality* can

help to pinpoint key regulators of network function [25]. Within TargetMine we have classified genes/proteins as network "hubs," that is, nodes with high *node degree* (many PPIs) that are believed to have the ability to influence network functions via multiple PPIs; and network "bottlenecks," that is, nodes with high *betweenness centrality* that are believed to represent central points for communication within the network [26]. The users can identify hubs and bottlenecks (predefined as the top 10% of the nodes in a species-specific PPI network ranked by "node degree distribution" and "betweenness" measures, respectively) in their list by expanding the "Network Properties" class in the "Model browser" and selecting the properties and attributes they wish to constrain upon their gene list. The users may also use one of the available templates such as "Gene(s) → PPI Network properties (Bottleneck and Hub)" to query for hubs and/or bottlenecks in their gene list.

(Note: Network topology parameters were calculated with HCDPs only)

## 6 Orthologue Mapping for Expanding Functional Annotations

1. Despite the advances in characterization of gene functions, most genome annotations are incomplete and they remain biased toward well-studied genes and proteins. Therefore, a large number of genes are still underrepresented in functional annotation databases or are bereft of any biological annotation altogether [27].
2. In TargetMine we have included mappings across orthologous genes in human, mouse and rat genomes. Orthologous genes are descended from a common ancestral gene as a consequence of speciation and thereby believed to retain similarity in structure and function including PPIs [28–30]. Orthology, therefore, forms an important basis for the transfer of functional annotation across genes and proteins [31].
3. TargetMine users can circumvent the limited species-specific information by using the single-click gene conversion system to transform a list of genes from one organism, say mouse, to a list of orthologous genes in the human genome. The "Convert to a different type" feature displayed as a top-right panel in the TargetMine "List Analysis" web page allows the users to quickly view and transform the items within the user-supplied list to another item type, for instance gene to protein and probesets to genes. The bottom half of the panel is subtitled "Orthologues," and it summarizes the number of orthologues (in parentheses) that were mapped in the corresponding species (Fig. 9).

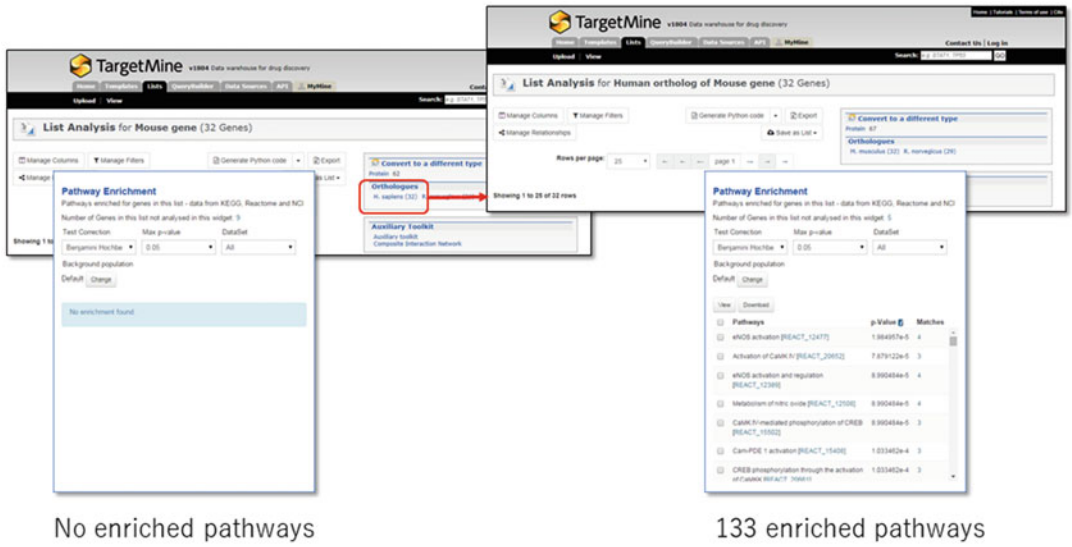


**Fig. 9** Orthologue conversion can help circumvent the paucity of functional annotations. In this example, the initial set of 32 mouse genes were not mapped to any enriched pathways. By transforming this gene list to a set of 32 human orthologous genes, 133 enriched pathways were identified

4. Clicking on the <species short name> (# of orthologues) hyperlink will direct the user to a new "List Analysis" web page titled "[] orthologues for the Gene list <> (# genes)", where "[]" refers to the corresponding organism, "<>" is the name of the initial user-supplied item list and "(# genes)" corresponds to the number of orthologous genes that were mapped and compiled in the new list (Fig. 9).
5. The users can then examine the enriched biological themes associated with the new list and also perform the requisite analysis using the new list (Fig. 9).

## 7 TargetMine Auxiliary Toolkit for Streamlining Data Analysis and Visualization

TargetMine Auxiliary Toolkit [6] is a suite of programs that features a series of interactive and easy-to-use tools to query the data in TargetMine, perform data analysis, define a data analysis workflow, and visualize the results without any scripting or programming efforts on the part of the user. The auxiliary toolkit complements TargetMine in two major ways: (a) *biological enrichment analysis* with data conversion, graphs and charts, including *association heat maps* and (b) *composite interaction network* for network visualization and analysis.

Detailed information about the auxiliary toolkit can be readily accessed by clicking the "TargetMine Auxiliary Toolkit" button on the TargetMine platform home page (http://targetmine.

mizuguchilab.org/) (Fig. 1). The individual analytical features can be accessed by clicking on the corresponding buttons (see below) on the home page or from the "Auxiliary Toolkit" panel in the "List Analysis" web page in TargetMine. Below we describe in detail how to use the different features within the auxiliary toolkit and how these implementations can enable the users to perform complicated queries with only a few clicks of the mouse that would otherwise necessitate multiple steps and invoking multiple queries in TargetMine.

1. *Enrichment analysis* The augmented enrichment analysis feature (along with the "Composite interaction network" feature; see below) can be individually accessed from the TargetMine home page (http://targetmine.mizuguchilab.org/; Fig. 1) or by using the "Auxiliary toolkit" link from the "Auxiliary Toolkit" in the TargetMine Lists page. Clicking the "Auxiliary toolkit" link on the home page will direct the user to a designated input form, where the user can type/paste a list of gene IDs in the "Type/Paste in identifiers;" box, select the organism from the drop-down menu next to "Organism:" or upload a gene list from file by clicking the "Browse" button followed by clicking "Read" button. The user can also import a gene list from TargetMine by clicking the "Import" button (this process requires TargetMine web service). Finally, the user can click the "GO, Create list" button to submit the list of genes for subsequent analysis. The user is then directed to the "TargetMine Auxiliary Toolkit" web page, where the items in the user-supplied list are displayed as a table of objects with accompanying information. The "Auxiliary toolkit" link in the Lists page directly transfers the user to "TargetMine Auxiliary Toolkit" page from TargetMine. To the right of the table are three panels that allow the users to perform "Enrichment Analysis with Charts and Graphs," "Convert to Orthologous Genes," and "Add Interacting Partners into the List." We discuss these features individually below (Fig. 10).
   (a) *Enrichment Analysis with Charts and Graphs* The "Enrichment Analysis with Charts and Graphs" panel allows the user to perform functional enrichment analysis of the input gene list by clicking the "Analysis" button. The user is then directed to the "Enrichment Analysis" page. The "Enrichment Analysis" allows the user to specify the biological theme type and dataset for enrichment analysis and select the multiple-test correction procedure from the drop corresponding drop-down menus. The user can also enter a value in the box to the right of "Filtered with p-value $\leq$" and then click the "Apply" button to specify the $p$-value threshold. The enriched biological themes are graphically summarized as "Bar


**Fig. 10** Enrichment analysis with TargetMine auxiliary toolkit. The user is required to specify the organism and either type/paste common gene

chart of the enriched <biological theme>", displaying the top ten enriched themes; the bar chart plots a histogram of the abundance of the genes/items mapped to a specific enriched theme in the user-supplied list along with their relative abundance in the background population; these values are reflected in percentage terms. Below the graph there is a table of enriched biological themes (database identifier and name) along with their number of matches in the list and the background population and the associated *p*-values. By default, only the top ten enriched themes are displayed in the table. The user can select the number of rows to be displayed in the table by selecting from the drop-down menu next to "Rows per page." The user can scroll through all the items in the table by clicking on the "►" icon below the table, the users can go back to the previous items by clicking on the "◄" icon. The user can view the selected gene–biological theme associations as a heat map by selecting checkboxes next to the entries in the display (or the checkbox on the top of the table to view all of them) and then clicking the "Show HeatMap" button (see below) (Fig. 10).

(b) *Orthologue conversion* The "Convert to Orthologous Genes" panel allows the user to convert their genes from one organism, to a list of orthologous genes in another organism (only human, mouse and rat genes can be interconverted in this manner) by selecting the organism from the drop-down menu next to "Convert to" function, followed by clicking the "Convert" button (Fig. 10).

(c) *Extending the user-supplied gene list* The users can "extend" the query gene list by including PPI partners by using the "Add Interacting Partners into the List" feature. The user can choose "All" (unfiltered), HC, or HCDP from the "interactions" drop-down menu. The user can choose to filter newly added genes by their cell/

**Fig. 10** (continued) identifiers in the corresponding box or browse to upload a list from a local file, click "Read" to read the contents of the file and then click the "GO, Create List" button. On the next page, the user can view a summary of the functional annotations with the query list and exclude specific genes from further analysis by clicking the "Remove" button. On the next page, the user can select datasets for enrichment analysis, customize *p*-value thresholds, export results to a file, and generate a heat map of the results by selecting the checkboxes against the enriched associations and clicking the "Show HeatMap" button. On the heat map page, the user can export the heat map as an image or the underlying matrix as a .txt file, reorder the heat map by gene or biological theme clusters and examine clusters of functionally correlated genes

tissue-specific expression by clicking the "add tissue constraints" link. The user will be prompted to enter a keyword for the cell tissue in the "Search" box and a list of cell/tissue types related to the query will automatically appear in the "Available items:" box. The user can click to select the individual cell/tissue types and transfer them to the "Selected items:" box by clicking the "→" button between the two boxes. Conversely, the user can remove the entries from the "Selected items:" by clicking the individual entries followed by clicking the "←" button and transfer them back to the "Available items:" box. The users can finalize their constraints by clicking the "OK" button at the bottom of the panel. The users may also upload a list of cell/tissue types by clicking the "Upload a list" and typing or pasting cell/tissue types in the resulting box and clicking "OK." Finally, the user can create an extended gene list by clicking the "Extend list" button for subsequent operations and enrichment analyses (Fig. 10).

2. *Association Heat Map* To assist with the visualization and analysis of multiple associations between genes and the associated biological themes, we have introduced the *Association Heat Map* function in the auxiliary toolkit. The user can visualize a heat map of enriched biological theme associations by clicking the "Show HeatMap" button in the "Enrichment Analysis" page. The user is directed to the "Gene-<Biological theme> Association Heat Map" page to visualize the selected associations as a two-way hierarchically organized mosaic plot (genes/items as rows and biological themes as columns) such that the genes that share a greater proportion of enriched associations are clustered as bright spots on the grid. The mosaic plot is bound by dendrograms on the top and left displaying the hierarchical organization of the association clusters that allows the users to identify clusters of functionally correlated genes as well as outliers that share little functional overlap with other items in the user-supplied list. The users can reshape the grid by rearranging the column order (by gene cluster or gene names) and/or the row order (by biological theme name or cluster) by selecting the option from the drop-down menus next to "Column Order:" and "Row Order:" respectively. The users specify the colors in the grid by clicking on the "Change color" button and then either selecting from the preexisting list of colors or entering the hex color code in the boxes for "Yes" and "No" values. The users can export the heat map as a publication quality image by clicking the "Export image" button and also export the underlying matrix as a text document by clicking the "Export matrix" button (Fig. 10). Alternatively, the users can

also paste or upload their own gene–biological theme associations as column-delimited values, such as those exported from the results of a TargetMine query in the "Association Heat Map" designated input form that is accessible from the "TargetMine Auxiliary Toolkit" button in the home page. Next, the users are directed to a "Preview" page, where they are prompted to select two columns, the values of which will be used to assemble the heat map, using the drop-down menus beside "Column 1" and "Column 2," respectively. Finally, the users can click the "Draw HeatMap" button to visualize the user-supplied associations as a heat map.

3. *Composite interaction network* Cellular networks are made up of multiple organizational layers that involve different types of biomolecular interactions such as PPIs, microRNA (miRNA)–-target interactions (MTIs), transcription factor (TF)–target gene interactions, and PCIs. A fundamental approach in systems biology is to understand how these components come together and define the complexity and functioning of all biological processes. The auxiliary toolkit permits the users to construct and visualize a *composite interaction network* of all the biomolecular interactions that are associated with the user-supplied gene list. This feature can be accessed by clicking the "Composite network" button on the TargetMine home page. The user is then transferred to the designated input form in the "Composite Interaction Network" page; the user can type/paste a list of gene IDs in the "Type/Paste in identifiers;" box and select the organism from the drop-down menu above. The right side of the page allows the user to select the interaction types to construct the interaction network by selecting the checkboxes next to individual interaction types. When selecting PCIs, an additional menu appears, where the users can choose to filter PCIs by affinity, that is, on the basis of IC50, Ki, or Kd values (in nm) that can be typed in the box next to "<Affinity parameter type>≤" tag. Finally, the user can click the "GO, Create List" button to submit the list of genes for building the composite network and the user is transferred to the "Composite Interaction Network" page (Fig. 11). The "Composite interaction network" in the TargetMine Lists page also transfers the user to "Composite Interaction Network" page. The left panel contains a table of user selected genes and basic accompanying information (Gene ID, symbol, and name). The user can remove individual genes from the list by clicking on the "Remove" button associated with each gene (the remove action can be undone by clicking on the "Undo remove" button at the bottom). The right of the table allows the user to reassess their selection of the interaction types and make any changes by selecting or unselecting the checkboxes

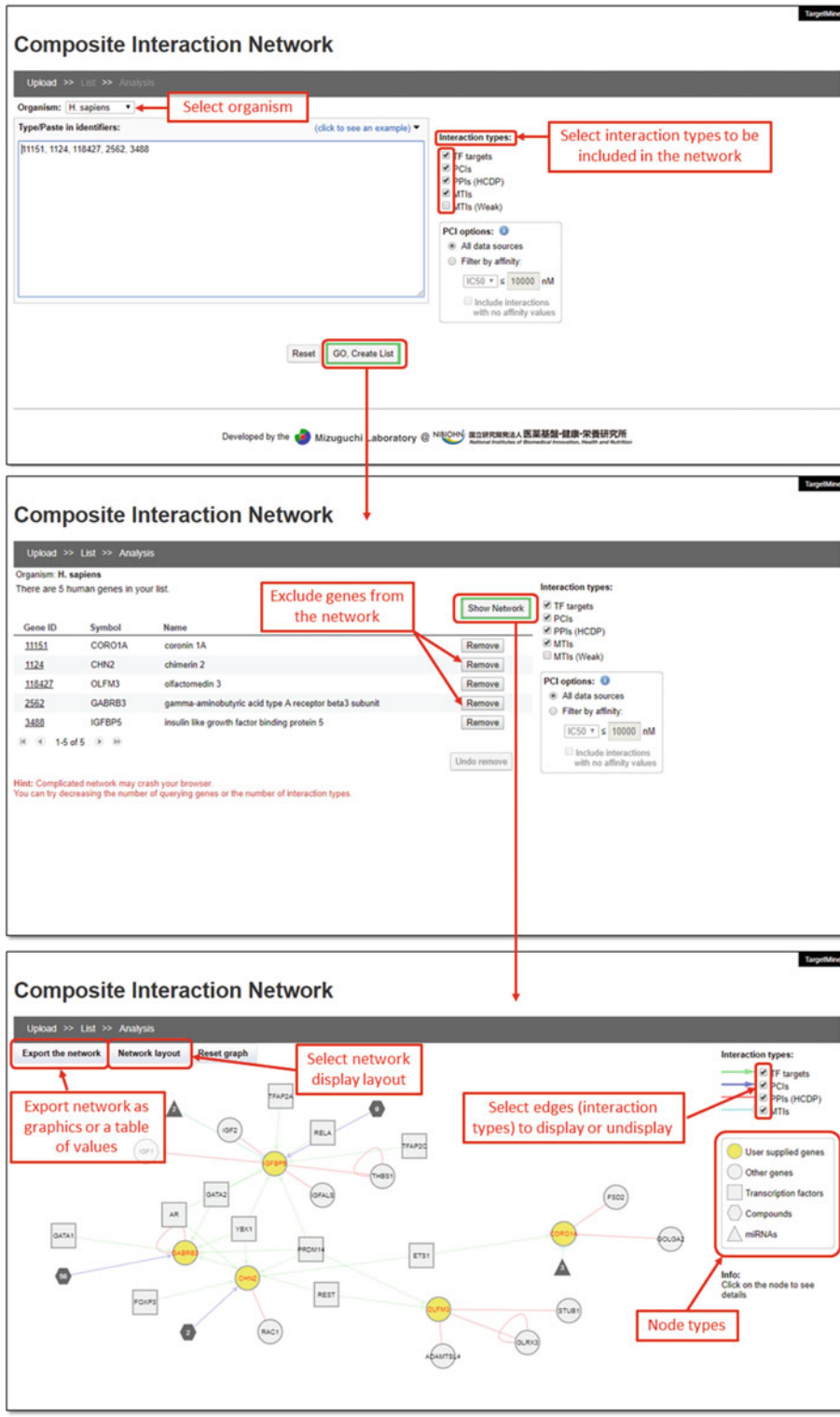


**Fig. 11** Constructing a composite interaction network with TargetMine auxiliary toolkit. The user is required to specify the organism and either type/paste common gene identifiers in the corresponding box or browse to

next to individual interaction types (Fig. 11). Finally, the user can visualize the composite network by clicking the "Show Network" button above the gene lists table. The user is transferred to a new page that displays the composite network constructed with the user-supplied genes and their interacting partners. The constituent biomolecules and interaction types are indicated by different colors and shapes that are summarized by a key on the right side of the page. The user can display (or undisplay) the individual interaction types by selecting or unselecting the corresponding checkboxes; select the network layout and zoom in and zoom out to visualize the network. The user can click the individual nodes and edges for more information about them. The users can export the network as image files or as a GraphML/tsv/XGMML file for visualization in stand-alone tools (Fig. 11).

## 8 A Suggested Protocol for Gene Set Analysis with TargetMine

Here we propose a simple and easy-to-use protocol for users to getting started with gene set analysis with TargetMine (Fig. 12).

1. As the first step in gene set analysis, the user can upload a set of initial candidate genes or proteins (e.g., a set of DEGs [or probes] derived from a microarray experiment or a set of proteins that interact with a given protein or set of proteins) to TargetMine and create a list.
2. In the next step the user can gather the genes mapped to the top $N$ significant associations (see above; where $N = 1,2,3\ldots$) retrieved from one or more of the enrichment widgets. In the event of two or more lists, the user can merge them and perform an intersect operation using the list function to summarize the enriched functional associations and obtain a list of prioritized genes.
3. If the enrichment analysis fails to return the desired significantly enriched annotations, the user can retrieve the HCDPs (and/or other interaction types) for their initial gene list and merge them to create an extended gene list (see above); the user can then reperform the enrichment analysis and process the enrichment analysis results as described in **step 2**. The user

**Fig. 11** (continued) upload a list from a local file, click "Read" to read the contents of the file; the user can also specify the interaction types for inclusion into the network by selecting/unselecting the corresponding checkboxes and then click the "GO, Create List" button. On the next page, the user can exclude specific genes from further analysis by clicking the "Remove" button and then click the "Show Network" button to create and visualize the composite interaction network. Next, the user can export the network as graphics or a table of values, change the network display layout, and also display or undisplay interaction types from the final view

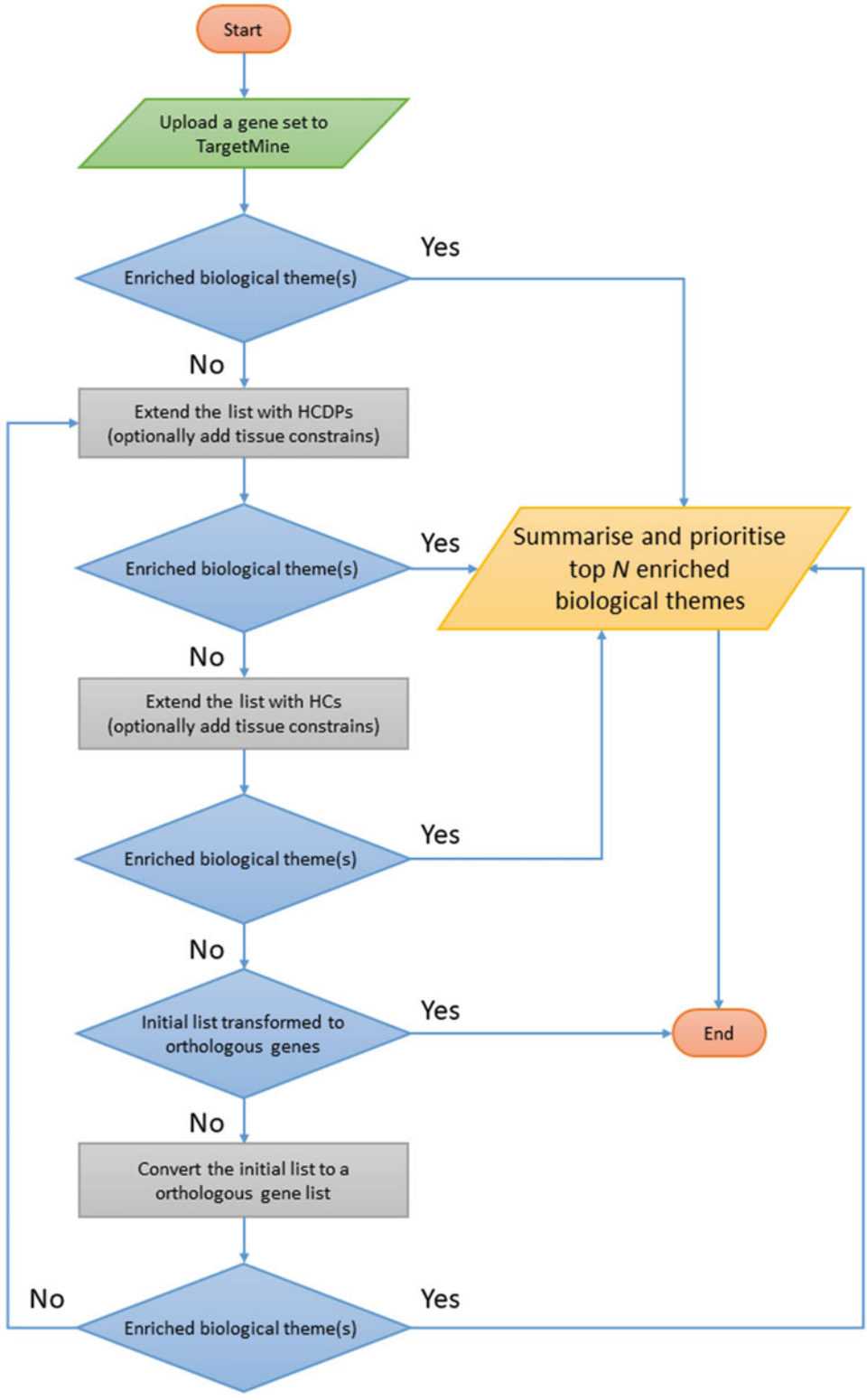


**Fig. 12** A suggested protocol for gene set analysis using TargetMine

can also filter HCDPs by gene expression. For instance, if the user-supplied gene list was derived from a microarray experiment that profiled liver cells, the users can specify that the interacting genes that are incorporated into the extended gene list must be highly expressed in liver and liver-associated cell types (see above) as well. The extended gene list may also be used for the construction of context-specific gene–gene interaction networks.

4. In the event the extended gene list too fails to turn up significantly enriched associations, the user may use the more relaxed HC dataset for inclusion into the extended gene list and proceed as in **step 3**. However, a dataset consisting of HCs rather than HCDPs may not be used reliably for constructing binary gene–gene interaction networks.
5. If **steps 1–4** fail to return any significantly enriched biological associations, the user may perform orthologue conversion to transform the initial gene list into orthologous genes and repeat the **steps 1–4** with the orthologous gene set.

## References


1. Ritchie MD, Holzinger ER, Li R et al (2015) Methods of integrating data to uncover genotype-phenotype interactions. Nat Rev Genet 16(2):85–97. https://doi.org/10.1038/nrg3868
2. Stein LD (2003) Integrating biological databases. Nat Rev Genet 4(5):337–345. https://doi.org/10.1038/nrg1065; pii: nrg1065
3. Triplet T, Butler G (2014) A review of genomic data warehousing systems. Brief Bioinform 15 (4):471–483. https://doi.org/10.1093/bib/bbt031
4. Wong L (2002) Technologies for integrating biological data. Brief Bioinform 3(4):389–404
5. Chen YA, Tripathi LP, Mizuguchi K (2011) TargetMine, an integrated data warehouse for candidate gene prioritisation and target discovery. PLoS One 6(3):e17844. https://doi.org/10.1371/journal.pone.0017844
6. Chen YA, Tripathi LP, Mizuguchi K (2016) An integrative data analysis platform for gene set analysis and knowledge discovery in a data warehouse framework. Database (Oxford) 2016. https://doi.org/10.1093/database/baw009
7. Smith RN, Aleksic J, Butano D et al (2012) InterMine: a flexible data warehouse system for the integration and analysis of heterogeneous biological data. Bioinformatics 28 (23):3163–3165. https://doi.org/10.1093/bioinformatics/bts577
8. Hamano Y, Kida H, Ihara S et al (2017) Classification of idiopathic interstitial pneumonias using anti-myxovirus resistance-protein 1 autoantibody. Sci Rep 7:43201. https://doi.org/10.1038/srep43201
9. Ihara S, Kida H, Arase H et al (2012) Inhibitory roles of signal transducer and activator of transcription 3 in antitumor immunity during carcinogen-induced lung tumorigenesis. Cancer Res 72(12):2990–2999. https://doi.org/10.1158/0008-5472.CAN-11-4062
10. Jin Y, Takeda Y, Kondo Y et al (2018) Double deletion of tetraspanins CD9 and CD81 in mice leads to a syndrome resembling accelerated aging. Sci Rep 8(1):5145. https://doi.org/10.1038/s41598-018-23338-x
11. Tripathi LP, Kambara H, Chen YA et al (2013) Understanding the biological context of NS5A-host interactions in HCV infection: a network-based approach. J Proteome Res 12 (6):2537–2551. https://doi.org/10.1021/pr3011217
12. Tripathi LP, Kambara H, Moriishi K et al (2012) Proteomic analysis of hepatitis C virus (HCV) core protein transfection and host regulator PA28gamma knockout in HCV pathogenesis: a network-based study. J Proteome Res 11(7):3664–3679. https://doi.org/10.1021/pr300121a
13. Tripathi LP, Kataoka C, Taguwa S et al (2010) Network based analysis of hepatitis C virus core

and NS4B protein interactions. Mol BioSyst 6 (12):2539–2553. https://doi.org/10.1039/c0mb00103a

14. Chen YA, Tripathi LP, Dessailly BH et al (2014) Integrated pathway clusters with coherent biological themes for target prioritisation. PLoS One 9(6):e99030. https://doi.org/10.1371/journal.pone.0099030
15. Ashburner M, Ball CA, Blake JA et al (2000) Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nat Genet 25(1):25–29. https://doi.org/10.1038/75556
16. Aoki-Kinoshita KF, Kanehisa M (2007) Gene annotation and pathway mapping in KEGG. Methods Mol Biol 396:71–91. https://doi.org/10.1007/978-1-59745-515-2_6
17. Matthews L, Gopinath G, Gillespie M et al (2009) Reactome knowledgebase of human biological pathways and processes. Nucleic Acids Res 37(Database issue):D619–D622. https://doi.org/10.1093/nar/gkn863
18. Schaefer CF, Anthony K, Krupa S et al (2009) PID: the Pathway Interaction Database. Nucleic Acids Res 37(Database issue): D674–D679. https://doi.org/10.1093/nar/gkn653
19. Afgan E, Baker D, van den Beek M et al (2016) The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. Nucleic Acids Res 44(W1): W3–W10. https://doi.org/10.1093/nar/gkw343
20. Benjamini Y, Drai D, Elmer G et al (2001) Controlling the false discovery rate in behavior genetics research. Behav Brain Res 125 (1–2):279–284
21. Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate - a practical and powerful approach to multiple testing. J R Stat Soc B 57(1):289–300
22. Dunn OJ (1961) Multiple comparisons among means. J Am Stat Assoc 56(293):52–64. https://doi.org/10.1080/01621459.1961.10482090
23. Holm S (1979) A simple sequentially rejective multiple test procedure. Scand J Stat 6 (2):65–70
24. Cline MS, Smoot M, Cerami E et al (2007) Integration of biological networks and gene expression data using Cytoscape. Nat Protoc 2(10):2366–2382. https://doi.org/10.1038/nprot.2007.324; pii: nprot.2007.324
25. Raman K (2010) Construction and analysis of protein-protein interaction networks. Autom Exp 2(1):2. https://doi.org/10.1186/1759-4499-2-2
26. Yu H, Kim PM, Sprecher E et al (2007) The importance of bottlenecks in protein networks: correlation with gene essentiality and expression dynamics. PLoS Comput Biol 3(4):e59. https://doi.org/10.1371/journal.pcbi.0030059
27. Edwards AM, Isserlin R, Bader GD et al (2011) Too many roads not taken. Nature 470 (7333):163–165. https://doi.org/10.1038/470163a
28. Fitch WM (2000) Homology a personal view on some of the problems. Trends Genet 16 (5):227–231
29. Koonin EV (2005) Orthologs, paralogs, and evolutionary genomics. Annu Rev Genet 39:309–338
30. Webber C, Ponting CP (2004) Genes and homology. Curr Biol 14(9):R332–R333
31. Watson JD, Laskowski RA, Thornton JM (2005) Predicting protein function from sequence and structural data. Curr Opin Struct Biol 15(3):275–284. https://doi.org/10.1016/j.sbi.2005.04.003; pii: S0959-440X(05)00082-5

# Chapter 4

# A Review of Microarray Datasets: Where to Find Them and Specific Characteristics

**Amparo Alonso-Betanzos, Verónica Bolón-Canedo, Laura Morán-Fernández, and Noelia Sánchez-Maroño**

## Abstract

The advent of DNA microarray datasets has stimulated a new line of research both in bioinformatics and in machine learning. This type of data is used to collect information from tissue and cell samples regarding gene expression differences that could be useful for disease diagnosis or for distinguishing specific types of tumor. Microarray data classification is a difficult challenge for machine learning researchers due to its high number of features and the small sample sizes. This chapter is devoted to reviewing the microarray databases most frequently used in the literature. We also make the interested reader aware of the problematic of data characteristics in this domain, such as the imbalance of the data, their complexity, and the so-called dataset shift.

**Key words** Microarray data, High dimensionality, Unbalanced data, Dataset shift

## 1 Introduction

All cells have a nucleus, and inside this nucleus there is DNA, which encodes the "program" for future organisms. DNA has coding and non-coding segments. The coding segments, also known as genes, specify the structure of proteins, which do the essential work in every organism. Genes make proteins in two steps: DNA is transcribed into mRNA and then mRNA is translated into proteins. Advances in molecular genetics technologies, such as DNA microarrays, allow us to obtain a global view of the cell, with which it is possible to measure the simultaneous expression of tens of thousands of genes [1]. Figure 1 displays the general process of acquiring the gene expression data from a DNA microarray. These gene expression profiles can be used as inputs to large-scale data analysis, for example, to increase our understanding of normal and diseased states.

During the last two decades, the advent of microarray datasets has stimulated a new line of research both in bioinformatics and in

Verónica Bolón-Canedo and Amparo Alonso-Betanzos (eds.), *Microarray Bioinformatics*, Methods in Molecular Biology, vol. 1986, https://doi.org/10.1007/978-1-4939-9442-7_4, 

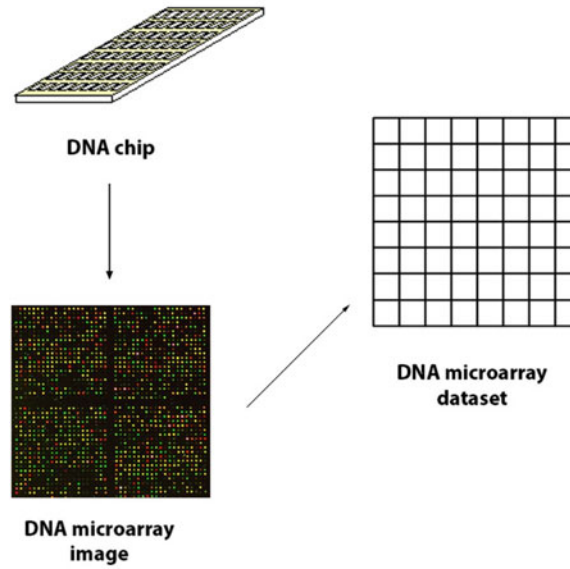


**Fig. 1** General process of acquiring the gene expression data from DNA microarray

machine learning. Although there are usually very small samples (often less than 100 patients) for training and testing, the number of features in the raw data ranges from 6000 to 60,000, since it measures the gene expression en masse. A typical classification task is to separate healthy patients from cancer patients based on their gene expression "profile" (binary approach). There are also datasets in which the goal is to distinguish among different types of tumors (multiclass approach), making the task even more complicated.

Therefore, microarray data pose a serious challenge for machine learning researchers. Having so many fields relative to so few samples creates a high likelihood of finding "false positives" due to chance (both in finding relevant genes and in building predictive models) [1]. It becomes necessary to find robust methods to validate the models and assess their likelihood. Furthermore, additional experimental complications (like noise and variability) render the analysis of microarray data an exciting domain [2].

Several studies have shown that most genes measured in a DNA microarray experiment are not relevant in the accurate classification of different classes of the problem [3]. To avoid the problem of the "curse of dimensionality" [4], feature (gene) selection plays a crucial role in DNA microarray analysis, which is defined as the process of identifying and removing irrelevant features from the training data, so that the learning algorithm focuses only on those aspects of the training data useful for analysis and future prediction [5].

Apart from the large number of genes vs small sample size, microarray data have other particularities such as the imbalance of

the data, their complexity, the presence of overlapping and outliers, or the so-called dataset shift. These problematics render the analysis of microarray data an interesting domain.

The rest of this chapter will comment on the specific characteristics of microarray data as well as providing a summary of the characteristics of the most famous datasets used in the literature and existing repositories.

## 2 Microarray Datasets

This section will be focused on where to find and where have been used microarray datasets. First, Subheading 2.1 will enumerate the existing microarray data repositories, while Subheading 2.2 provides a summary of the characteristics of the most famous binary and multiclass datasets used in the literature.

### 2.1 DNA Microarray Repositories

Although in the initial development of DNA microarray data analysis it was difficult to find datasets to deal with, in recent years there have been a growing number of public microarray data repositories of a wide spectrum of cancer types available for the scientific community. There are some websites specialized in this type of datasets, the most famous are listed below:

- *ArrayExpress*, from the European Bioinformatics Institute [6]:
  http://www.ebi.ac.uk/arrayexpress/
- *Gene Expression Omnibus*, from the National Institutes of Health [7]:
  http://www.ncbi.nlm.nih.gov/geo/
- *The Cancer Genome Atlas (TCGA)*, from both the National Cancer Institute and the National Human Genome Research Institute [8]:
  https://cancergenome.nih.gov/
- *Cancer Program Datasets*, from the Broad Institute [9]:
  http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi
- *Dataset Repository*, from the Bioinformatics Research Group of Universidad Pablo de Olavide [10]:
  http://eps.upo.es/bigs/datasets.html
- *Gene Expression Model Selector*, from Vanderbilt University [11]:
  http://www.gems-system.org
- *Gene Expression Project*, from Princeton University [12]:
  http://genomics-pubs.princeton.edu/oncology/
- *TAIR Microarrays*, from The Arabidopsis Information Resource [13]:
  https://www.arabidopsis.org/portals/expression/microarray/

- Genevestigator commercialized by NEBION AG under exclusive license from the Swiss Federal Institute of Technology Zurich (ETH) [14]

  https://genevestigator.com
- AROMA: An open-source R framework for your microarray analysis [15]:

  http://www.aroma-project.org/
- *ELVIRA Biomedical Dataset Repository* [16]

  http://leo.ugr.es/elvira/DBCRepository/

In addition, there are data repositories from the field of machine learning that contain many of these datasets, it is worth mentioning:

- *Machine learning dataset repository*, supported by Pattern Analysis, Statistical Modeling, and Computational Learning (PASCAL) [17]:

  http://mldata.org/repository/data/
- *Kaggle.The Home of Data Science & Machine Learning* [18]:

  https://www.kaggle.com/datasets
- *UCI Machine Learning Repository* from the Center for Machine Learning and Intelligent Systems, University of California (Irvine) [19]:

  http://archive.ics.uci.edu/ml/index.php
- *Feature Selection Datasets*, from Arizona State University [20]:

  http://featureselection.asu.edu/datasets.php
- *Bioconductor* Open-source software for bioinformatics [21]:

  http://www.bioconductor.org/

### 2.2 Datasets

As mentioned in Subheading 1, there are two types of microarray datasets present in the literature. The most famous are the binary datasets, usually related to separating healthy patients from cancer patients. When the goal is to distinguish between different types of tumors, we can find multiclass datasets, in which the classification task becomes more complicated.

Table 1 displays binary microarray datasets used in the literature. There the number of samples *s*, the number of features *f*, the class distribution, the original reference of the dataset, the year when the dataset was published, the references of some works using the dataset, and where it is available for download can all be found. When a data is not available, it is represented as "*unknown*." In turn, Table 2 visualizes the multiclass microarray datasets. In this case, *c* stands for the number of classes and the class distribution is not shown due to the high diversity in the number of classes.

**Table 1**
**Dataset description for binary datasets: *s* and *f* are the number of samples and features, respectively**

| Dataset | *s* | *f* | Distribution | Original ref. | Year | Where used | Download |
|---|---|---|---|---|---|---|---|
| B_MD | 34 | 7129 | 26–74% | [22] | 2002 | [23] | *unknown* |
| Bone Lesion | 173 | 12,625 | *unknown* | [24] | 2003 | [23] | *unknown* |
| Brain | 21 | 12,625 | 33–67% | [25] | 2003 | [26–30] | [9] |
| Brain_Tumor1 | 60 | 7129 | *unknown* | *unknown* | *unknown* | [31] | *unknown* |
| Brain_Tumor2 | 50 | 12,625 | *unknown* | *unknown* | *unknown* | [31] | *unknown* |
| Breast | 22 | 3226 | *unknown* | [32] | 2001 | [33] | *unknown* |
| Breast Cancer | 97 | 24,481 | *unknown* | [34] | 2002 | [26, 35–38] | [16, 17] |
| Breast-test | 19 | 24,481 | 37–63% | [34] | 2002 | [36, 39] | [16] |
| Breast-train | 78 | 24,481 | 56–44% | [34] | 2002 | [29, 36, 39] | [16] |
| BreastER | 49 | 7129 | 49–51% | [40] | 2001 | [23] | *unknown* |
| BR-ER49 | 49 | 6817 | 49–51% | [40] | 2001 | [41] | *unknown* |
| C_MD | 60 | 7129 | 35–65% | [22] | 2002 | [23] | *unknown* |
| Celiac | 132 | 22,185 | *unknown* | [42] | 2009 | [23] | *unknown* |
| CNS/ Embrional-T | 60 | 7129 | 35–75% | [22] | 2002 | [26–29, 35, 36, 43, 43–47] | [9, 10] |
| Colon | 62 | 2000 | 35–65% | [48] | 1999 | [23, 33, 46, 49–52] [26–28, 35, 36, 44, 53] [29, 39, 41, 54–57] [30, 38, 47, 58, 59] | [9, 10, 12, 16, 17] |
| Colon-epi | 202 | 44,290 | *unknown* | [60] | 2008 | [23] | *unknown* |
| DLBCL | 77 | 5470 | 75–25% | [61] | 2002 | [47, 49, 56, 59, 62] | [11] |
| DLBCL | 47 | 4026 | 49–51% | [63] | 2000 | [26–29, 33, 35, 36, 43, 45] | [9, 17] |
| DLBCL | 77 | 7129 | 75–25% | [61] | 2002 | [31, 44, 46, 55] | [9, 17] |
| GLI-85 | 85 | 22,283 | 31–69% | [64] | 2004 | [26–29] | [20] |
| Leukemia/ ALLAML | 72 | 7129 | 35–65% | [3] | 1999 | [26, 35, 43, 44, 49, 53, 58] [23, 33, 37, 38, 41, 54, 59] [30, 46, 51, 65] | [9, 16, 17] |
| Leukemia_test | 34 | 7129 | 71–29% | [3] | 1999 | [36, 39, 57, 66] | [9, 10] |
| Leukemia_train | 38 | 7129 | 59–41% | [3] | 1999 | [36, 39, 57, 66, 67] | [9, 10] |
| Lung | 52 | 918 | 75–25% | [68] | 2001 | [23] | *unknown* |
| Lung | 181 | 12,533 | 83–17% | [69] | 2002 | [26, 35, 39, 43, 59, 70] | [9, 16] |

(continued)

**Table 1**
**(continued)**

| Dataset | *s* | *f* | Distribution | Original ref. | Year | Where used | Download |
|---|---|---|---|---|---|---|---|
| Lung | 410 | 2428 | 34–66% | [71] | 2008 | [58] | *unknown* |
| Lung_test | 149 | 12,533 | 90–10% | [69] | 2002 | [36, 39] | [9, 16] |
| Lung_train | 32 | 12,533 | 50–50% | [69] | 2002 | [36, 39] | [9, 16] |
| LUNG181 | 181 | 12,600 | 17–83% | [69] | 2002 | [41] | *unknown* |
| LYM77 | 77 | 6817 | 25–75% | [61] | 2002 | [41] | *unknown* |
| Lymphoma/ B-cell1 | 45 | 4026 | 49–51% | [63] | 2000 | [53, 57] | [9] |
| Moffitt colon cancer | 122 | 2619 | 31–69% | [72] | 2005 | [58] | *unknown* |
| Ovarian | 253 | 15,154 | 36–64% | [73] | 2002 | [26–30, 35–37, 43, 51] | [16, 17] |
| Prostate | 102 | 6033 | 51–49% | [74] | 2002 | [50, 53] | *unknown* |
| Prostate | 136 | 12,600 | 43–57% | [74] | 2002 | [26, 35, 43, 46, 59] | [9, 16] |
| Prostate-test | 34 | 12,600 | 26–74% | [74] | 2002 | [36, 39, 54, 75] | [16, 17] |
| Prostate-train | 102 | 12,600 | 49–51% | [74] | 2002 | [29, 31, 36, 39, 41, 54, 55] [45, 65, 75] | [16, 17] |
| Prostate Tumor | 102 | 10,509 | 51–49% | [74] | 2002 | [30, 47, 49, 62] | [11] |
| SMK-CAN-187 | 187 | 19,993 | 48–52% | [76] | 2007 | [26–29, 57, 66] | [20] |

## 3 Intrinsic Characteristics of Microarray Data

As mentioned in Subheading 1, microarray data classification poses a serious challenge for computational techniques, because of their large dimensionality (up to several tens of thousands of genes) with small sample sizes. Furthermore, there are additional experimental complications that render the analysis of microarray data an intriguing domain.

### *3.1 Small Sample Size*

The first problem that one may find when dealing with microarray data is related to the small sample size (usually less than 100), as microarrays are still costly, although the price has been diminishing progressively. Thus, microarray has increased the rate of data collection during the last years, but sample size is still a major issue when selecting features and building predictive models for medical applications. A key point in this regard is that error estimation is greatly impacted by small samples [94], as well as the low power for

**Table 2**
**Dataset description for multiclass datasets: *s*, *f*, and *c* are the number of samples, features, and classes, respectively**

| Dataset | *s* | *f* | *c* | Original ref. | Year | Where used | Download |
|---|---|---|---|---|---|---|---|
| 9-Tumors | 60 | 5726 | 9 | [77] | 2001 | [31, 47, 49, 62, 78] | [11] |
| 11-Tumors/ Carcinomas | 174 | 12,533 | 11 | [79] | 2001 | [31, 47, 49, 62, 65, 78, 80] | [11] |
| 14-Tumors | 308 | 15,009 | 26 | [81] | 2001 | [31, 47, 49, 62, 82] | [11] |
| Brain Tumor 1 | 90 | 5920 | 5 | [22] | 2002 | [31, 38, 47, 49, 62, 78] | [11] |
| Brain Tumor 2 | 50 | 10,367 | 4 | [25] | 2003 | [31, 49, 62, 78, 83] | [11] |
| CLL-SUB-111 | 111 | 11,340 | 3 | [84] | 2004 | [57, 66] | [20] |
| GCM | 198 | 16,306 | 14 | [81] | 2001 | [26, 33, 35, 36] | [9] |
| GCM | 190 | 16,063 | 14 | [81] | 2001 | [47, 80] | [10] |
| GLA-BRA-180 | 180 | 49,151 | 4 | [85] | 2006 | [26, 57] | [20] |
| Glioma | 50 | 12,625 | 4 | [25] | 2003 | [65] | [9] |
| Global Cancer Map/GCM-Train | 144 | 16,063 | 14 | [81] | 2001 | [57, 66] | [10] |
| Leukemia | 110 | 22,278 | *unknown* | *unknown* | *unknown* | [86] | *unknown* |
| Leukemia 1 | 72 | 5327 | 3 | [3] | 1999 | [31, 38, 47, 49, 62, 83] | [11] |
| Leukemia 2 | 72 | 11,225 | 3 | [87] | 2002 | [31, 38, 47, 49, 62] | [11] |
| Leukemia-MLL | 72 | 8359 | 3 | [87] | 2002 | [88] | [9] |
| Leukemia-MLL-train | 57 | 12,582 | 3 | [87] | 2002 | [75, 89] | [16, 17] |
| Leukemia-MLL-test | 15 | 12,582 | 3 | [87] | 2002 | [75] | [16, 17] |
| Lung | 254 | 8359 | 5 | [90] | 2001 | [88] | [9] |
| Lung-Cancer | 203 | 12,601 | 5 | [90] | 2001 | [31, 47, 49, 62, 65, 78, 89] | [11] |
| Lymphoma/B-cell3 | 96 | 4026 | 9 | [63] | 2000 | [26, 30, 35, 36, 49, 52, 57] | [11] |
| TOX-171 | 171 | 5748 | 4 | [91] | 2010 | [26, 57, 66] | [20] |
| SRBCT | 83 | 2309 | 4 | [92] | 2001 | [31, 33, 38, 49, 62, 70, 78, 88] | [11] |
| SRBCT-train | 63 | 2309 | 4 | [92] | 2001 | [75, 80] | [11] |
| SRBCT-test | 20 | 2309 | 4 | [92] | 2001 | [75] | [11] |
| Yeast | 79 | 2467 | *unknown* | [93] | 2000 | [51] | *unknown* |

statistical tests, as, for example, the widely used t-test for comparison of groups [95]. Regarding the validity of the statistical tests, several approaches have been proposed, mainly consisting in pooling of permutation-derived tests statistics across all genes [95–97]. The proposal in Yang and Churchill [95] demonstrated that using the standard *t*-test to define a subset of genes for pooling a threshold for subset feature selection can be determined, and using these provide correct type I error.[1] Murie et al. [98] offered a comparison in performance of statistical tests aimed at circumventing the problem of small sizes.

As said above, the number of datasets available has been growing during the last years, although the sample size of each dataset remains small regarding the number of features, as shown in Tables 1 and 2. One way of dealing with this problem was to combine multiple datasets [99, 100] that has the potential for increasing the power of microarray data analysis by pooling information, as just discussed. Combining datasets is however difficult, as on the one hand different normalization and summarization techniques are used. Also, due to the shortage of datasets measuring the same area, perhaps different platforms might be needed, complicating even more the problem. That is perhaps the main reason behind the fact that most studies are limited to single datasets of small size. But combination might also be achieved by using fold-change methods, as feature selection is improved that way, and feature selection is almost mandatory for these type of datasets. The first paper using finite mixture modeling and bootstrap inference to address the problems of false positive and false negative results was [101], but several others followed. For example, Phan et al. [102] proposed a wrapper-based selection technique that combines bootstrap estimated classification errors for individual genes across multiple datasets, reducing the contribution of datasets with high variance.

Without the appropriate estimation of the error, an unsound application of classification methods follows, which has generated a large number of publications and an equally large amount of unsubstantiated scientific hypotheses [103]. For example, in Michiels et al. [104] it is reported that reanalysis of data from the seven largest published microarray-based studies that have attempted to predict the prognosis of cancer patients reveals that five of those seven did not classify patients better than chance. To overcome this problem, it becomes necessary to select a correct validation method for estimating the classification error. Cross-validation error estimation is perhaps the most common way for estimating classification errors in microarrays, as they on average

[1] A type I error is the incorrect rejection of a true null hypothesis that usually has the effect of concluding that a given relationship exists when in fact it does not. That is, a type I error is a false positive.

nearly agree with the true errors. But the problem with cross-validation is that it might be highly affected by the small sample size. Braga-Neto and Dougherty [105] compared cross-validation, resubstitution, and bootstrap estimation using synthetic and real microarray data, concluding that although cross-validation is less biased, it has an excessive variance, and thus making individual estimates unreliable for small samples. Bootstrap methods, on the contrary, improve the performance relative to variance, but at a high computational cost and many times with increased bias. Finally, resubstitution tends to be low biased, even severely, although having the advantage of being computationally inexpensive and exhibiting lower variability than cross-validation. In Hanczar et al. [106] the conclusion on the lack of reliability in ROC performance results over microarrays due to small sample size is also relevant. Thus, these studies should be taken into account when using an error estimation method over small size microarray datasets. Alternatives have been proposed in Laber and Murphy [107], using confidence measures for the generalization error and providing a computationally efficient algorithm.

### 3.2 Class Imbalance

A common problem in real datasets is the so-called "class-imbalance problem." This occurs when a dataset is dominated by a major class or classes which have significantly more samples than the other rare/minority classes in the data [108–110]. In these cases, standard classification algorithms have a bias toward the classes with a greater number of instances, since the rules that correctly predict those instances are positively weighted in favor of the accuracy metric, whereas specific rules that predict examples from the minority class are usually ignored (treated as noise), because more general rules are preferred. This bias is even larger when data is high-dimensional, such as microarrays: the high dimensionality further increases the bias toward the classification into the majority class, even there is no real difference between the classes [111]. Therefore, minority class instances are more often misclassified than those from the other classes [112].

In the domain at hand, the cancer class tends to be rarer than the non-cancer class because usually there are more healthy patients. However, it is important for practitioners to predict and prevent the appearance of cancer. Examples of very unbalanced microarray datasets are Lung_test, or Brain Tumor 1, among others (Fig. 2). This problematic is of special importance when the imbalance is more marked in the test set than in the training set. Multiclass datasets also suffer from this problem: some type of tumors/tissues has fewer samples compared to others. For example, Brain Tumor 1 has 5 classes but the majority class takes 67% of the samples.

The traditional preprocessing techniques used to overcome this issue are undersampling and oversampling methods. Undersampling is a technique which creates a subset of the original dataset

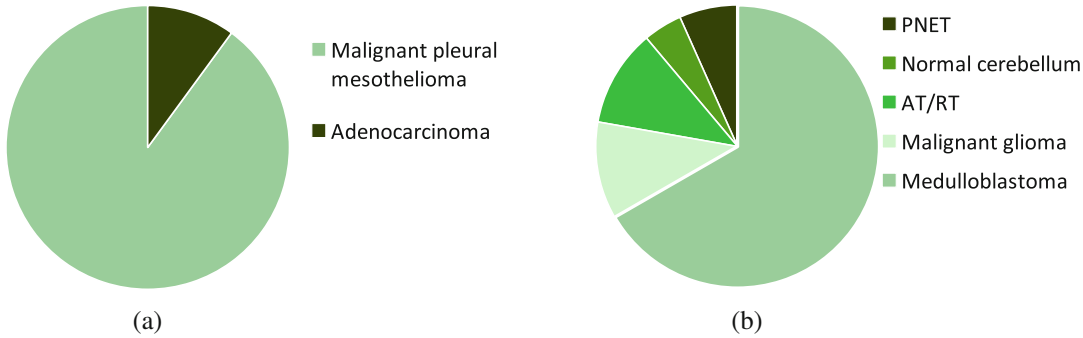


**Fig. 2** Class distributions for the (**a**) Lung_test and (**b**) Brain Tumor 1 datasets

by eliminating samples. It aims to attain the same number of samples of the majority class as in the minority class. As a very small number of samples are available in the microarray datasets, elimination of observations is not a good option. In contrast, oversampling methods create a superset of the original dataset by replicating some instances or creating new instances from existing ones. One of the most employed oversampling techniques is the so-called SMOTE [113], in which the minority class is oversampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the $k$ minority class nearest neighbors. This technique was applied in Blagus and Lusa [114] on microarray data, although the authors stated that it does not attenuate the bias toward classification in the majority class for most classifiers. Morán-Fernández et al. [115] observed that the imbalance ratio is not enough to predict the adequate performance of the classifier. As an alternative approach, authors computed several data complexity measures over the imbalance microarray datasets in order to support the application or not of an oversampling method. They recommend to analyze the theoretical complexity before applying the SMOTE algorithm since the classifier is more affected by the complexity of the microarray data itself than by the imbalance problem.

In recent years, ensemble of classifiers has arisen as a possible solution to the class-imbalance problem, attracting great interest among researchers [112, 116], in several cases combined with preprocessing techniques such as SMOTE. Ensemble-based algorithms have been proven to improve the results that are obtained by the usage of data preprocessing techniques and training a single classifier [112]. For all these reasons, it is worth considering this problematic when dealing with unbalanced microarray datasets. Another approach to deal with imbalanced microarray datasets could be one-class SVM trained only with the target class, which may lead to a better predictive performance [117, 118].

### 3.3 Data Complexity

Classical classifiers may fail to obtain acceptable accuracies on microarray data. In several cases drawbacks in the classifier performance could arise not because of deficiencies in the algorithms, but due to characteristics intrinsic to the data. On such a context, data complexity measures were proposed to represent data particularities that add complexity to classification tasks, such as overlaps between classes, separability and decision boundary linearity, and at identifying relationships with classification performance. The most commonly employed data complexity measures are those proposed by Ho and Basu [119], gathering metrics of three types:

1. Measures of overlap in feature values from different classes:
   - F1 returns the maximum Fisher's discriminant ratio over all features. Small values represent strong overlapping.
   - F2 is defined as the amount of overlap between bounding boxes of two classes, and it is zero if there is at least one feature in which the values of the two classes do not overlap.
   - F3 returns the maximum (individual) feature efficiency, i.e., the largest fraction of points distinguishable with only feature. Small values of this measure indicate high overlap.
2. Measures of class separability:
   - L1 evaluates to what extent the training data is linearly separable. It returns the sum of the differences between a linear classifier predicted and the current class value. This measure is zero for a linearly separable problem.
   - L2 returns the error rate of the linear classifier defined for L1, measured using the training set.
   - N1 returns the fraction of points on the class boundary. For doing so, it constructs a class-blind minimum spanning tree over the full dataset, counting the number of points incident to an edge going across the two classes. The index thus reflects the fraction of such points over all points in the dataset. High values of this measure indicate smaller separation and a more difficult classification task.
   - N2 is the ratio of average intra-/interclass NN distance. For each instance, the Euclidean distances to its nearest neighbor from the same class and its nearest neighbor from the other class are calculated. Then, the result is the ratio of the sum of the intraclass distances to the sum of the interclass distances for each instance. High values indicate that samples from the same class are disperse.
   - N3 returns the error rate of the 1-NN classifier, estimated by leave-one-out cross-validation on the training set.

3. Measures of geometry, topology, and density of manifolds:
   - L3 makes use of the linear classifier defined for L1 and returns the error rate obtained on a test set created from the training set by linear interpolation between randomly drawn pairs of points from the same class. For linear classifiers and linearly separable problems, it measures alignment of the decision surface with the class boundary.
   - N4 creates a test set as proposed by L3 and returns the error for the 1-NN classifier.
   - T1 counts the number of spheres needed to cover each class, where each sphere is centered at a training point and grown to the maximum size before it touches another class. The count is then normalized by the total number of points. In a problem where each point is closet to points in the other class than points in its own class, each point is covered by a distinctive sphere of a small size, resulting in a high value for the measure.
   - T2 is the simple ratio between the number of points in the dataset and the number of feature dimensions.

In the area of microarray classification, we can find several works using these data complexity measures. Lorena et al. [120], who, using the Ho and Basu measures, investigated the particular characteristics of several microarrays datasets that most impacted on the prediction ability of support vector machine classifiers. Okun et al. [121] in a novel approach based on using an ensemble of *k*-NN classifiers also found positive dependence between complexity and error. Bolón-Canedo et al. [122] analyzed in depth the theoretical complexity of several binary microarray datasets and then connecting it with the empirical results obtained by four widely used classifiers. An extension for microarray datasets with multiple classes was presented in Morán-Fernández et al. [123]. Experimental results for 21 binary and multiclass datasets demonstrated that a correlation exists between the theoretical complexity of microarray data and the classification error rates. Thus, some interesting properties of the microarray datasets were identified from these works:

- Almost all the microarray datasets, according to the values of L1 and L2, were linearly separable. Thus, SVM with a linear kernel was a good option, in overall, reporting high classification accuracies.
- The correlation between different classifiers and T1 pointed out that the sparsity of microarray data is a major aspect for indicating the complexity of microarray classification.
- For N2, the microarray datasets lent to higher values. This indicates that instances from the same class were dispersed in

the feature space. Both separability measures, N2 and N1, offer interesting relations with the NN classifier.

- F2, which was zero for all microarray datasets, did not yield any information. A side effect of calculating the volume is that the value of the measures decreases greatly as dimensionality increases. Thus, it might be interesting to compute the sum (length of the overlap) in this type of datasets.
- F1 and F3 overlapping measures have demonstrated to be related to classification accuracy. As these measures increase in value, overlapping between classes is reduced so better classification results are expected.

### 3.4 Dataset Shift

Another common problem when datasets were originally divided to training and test sets is the so-called *dataset shift*. This occurs when the testing (unseen) data experience a phenomenon that leads to a change in the distribution of a single feature, a combination of features, or the class boundaries [124]. As a result, the common assumption that the training and testing data follow the same distributions is often violated in real-world applications and scenarios, which may hinder the process of feature selection and classification. For example, Lung, Leukemia, and Prostate datasets have separated training and test sets (*see* Table 1). In the case of Lung, there is a single feature (#1136) which can correctly classify all the samples in the training set, as shown in Fig. 3a, in which different colors and shapes stand for different classes and the dashed line shows a clear linear separation between them. However, the same feature is not that informative in the test set and the class is not linearly separable, as displayed in Fig. 3b. Furthermore, note that there is an enormous disparity in class distribution: 50–50% in the training set and 90–10% in the test set. A similar situation happens with Leukemia dataset (*see* Fig. 4) and feature #4847.

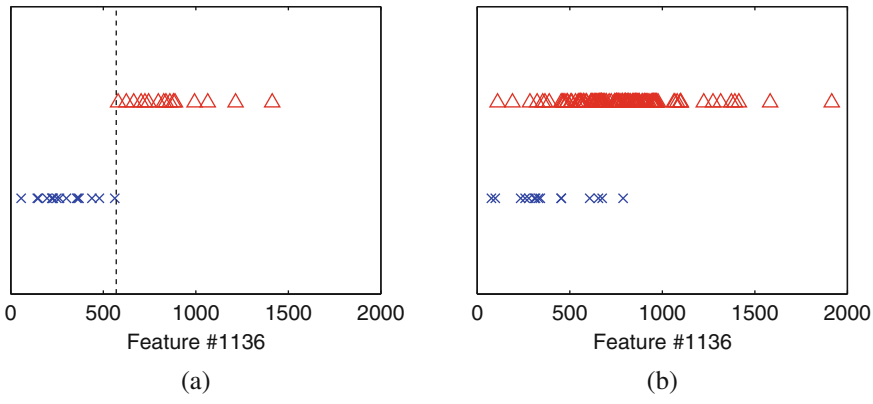


**Fig. 3** Feature #1136 in Lung dataset. (**a**) Lung_train. (**b**) Lung_test

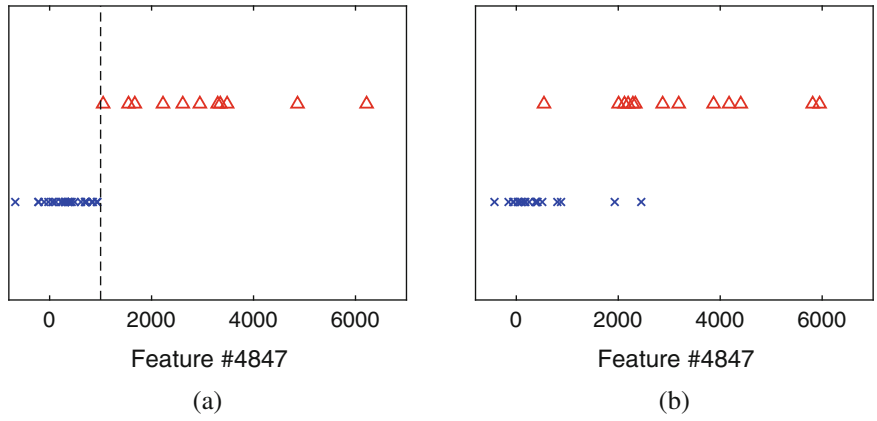


**Fig. 4** Feature #4847 in Leukemia dataset. (**a**) Leukemia_train. (**b**) Leukemia_test

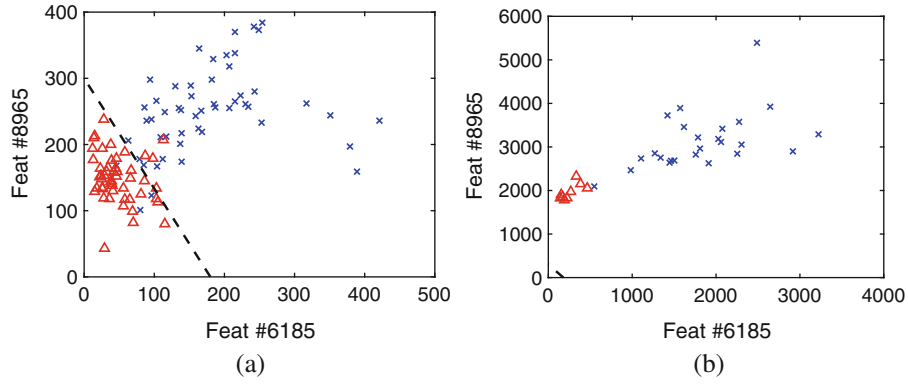


**Fig. 5** Features #6185 and #8965 in Prostate dataset. (**a**) Prostate-train. (**b**) Prostate-test

The Prostate dataset poses a big challenge for machine learning methods since the test dataset was extracted from a different experiment and has a nearly 10-fold difference in overall microarray intensity from the training data. In fact, the test distribution (26–74%) differs significantly from the train distribution (49–51%) and with an inappropriate or no feature selection, some classifiers simply assign all the samples to one of the classes [36, 43]. In fact, if we plot the two most relevant features according to their mutual information with the class (features #6185 and #8965), we can see in Fig. 5 some interesting facts. The two features range in very different intervals (notice that the maximum values in the $x$ axis are around 450 and 3500, respectively, for train and test). If we draw a line to separate the two classes in the training set, allowing some misclassifications, the same line in the test set is equivalent to assigning all the samples to one of the classes (see bottom left corner of Fig. 5b). More details about this phenomenon and a possible solution using normalization can be found in Bolón-Canedo et al. [122].

Dataset shift can also appear because of the validation technique chosen. One of the most widely used validation techniques in the microarray domain is the so-called *k-fold cross-validation*, along with its variant *leave-one-out cross-validation*. However, it has been shown [125] that cross-validation can potentially introduce dataset shift, a harmful factor that is often not taken into account and which can result in inaccurate performance estimation. To solve this problem, *distribution optimally balanced stratified cross-validation* (DOB-SCV) [125] is based on the idea that, by assigning close-by examples to different folds, each fold will end up with enough representatives of every region, thus avoiding dataset shift. To achieve this goal, DOB-SCV starts on a random unassigned example, and then finds its $k - 1$ nearest unassigned neighbors of the same class. Once it has found them, it assigns each of those examples to a different fold. The process is repeated until all examples have been assigned.

### 3.5 Outliers

An important aspect that has been neglected in the literature is to detect outliers [126] in microarray samples. In some microarray datasets, there are samples that are incorrectly labeled or identified as likely to be contaminated which should in fact be designated outliers, since they can exert a negative effect on the selection of informative genes for sample classification. In Kadota et al. [127], a method was developed which found some outlying samples in the well-known Colon dataset. Therefore, analysis of samples designated as outliers should be considered as a preprocessing step in the classification of microarray datasets because they can have a negative effect on the gene subset selection and, as a consequence, on the final prediction [128].

## 4 Summary

This chapter is a brief introduction to microarray datasets, describing the main reasons that have converted them to a challenging and fruitful field of application for machine learning algorithms. Due to their high dimensionality, dimensionality reduction techniques are to be used for correct classification and also for explainability of results to microarray experts, as many of the features of the datasets are irrelevant and/or redundant. For the interested reader, the most common repositories where one can download these type of datasets, both binary and multiclass, are given, together with descriptions of more than 50 specific microarray datasets. Finally, some insights on several characteristics of these datasets (such as small sample size, class imbalance, dataset shift, data complexity, and the existence of outliers) and their influence on the machine learning algorithms employed and the results obtained are given.

## References


1. Piatetsky-Shapiro G, Tamayo P (2003) Microarray data mining: facing the challenges. ACM SIGKDD Explor Newsl 5(2):1–5
2. Saeys Y, Inza I, Larrañaga P (2007) A review of feature selection techniques in bioinformatics. Bioinformatics 23(19):2507–2517
3. Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri MA *et al* (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286(5439):531–537
4. Jain A, Zongker D (1997) Feature selection: evaluation, application, and small sample performance. IEEE Trans Pattern Anal Mach Intell 19(2):153–158
5. Guyon I, Gunn S, Nikravesh M, Zadeh LA (2006) Feature extraction: foundations and applications, vol 207. Springer, Berlin
6. Arrayexpress - Functional Genomics Data (2018). http://www.ebi.ac.uk/arrayexpress/. [Online; accessed Jan 2018]
7. Gene Expression Omnibus (2018). http://www.ncbi.nlm.nih.gov/geo/. [Online; accessed Jan 2018]
8. The Cancer Genome Atlas (TCGA) (2018). https://cancergenome.nih.gov/. [Online; accessed Jan 2018]
9. Broad Institute (2018) Cancer Program Data Sets. http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi. [Online; accessed Jan 2018]
10. Dataset Repository, Bioinformatics Research Group (2018). http://www.upo.es/eps/bigs/datasets.html. [Online; accessed Jan 2018]
11. Statnikov A, Aliferis CF, Tsamardinos I (2018) Gems: gene expression model selector. http://www.gems-system.org. [Online; accessed Jan 2018]
12. Gene Expression Project (2014) Princeton University. http://genomics-pubs.princeton.edu/oncology/. [Online; accessed Jan 2014]
13. The Arabidopsis Information Resource, Gene Expression Resources (2018) https://www.arabidopsis.org/portals/expression/microarray/. [Online; accessed Jan 2018]
14. Hruz T, Laule O, Szabo G, Wessendorp F, Bleuler S, Oertle L, Widmayer P, Gruissem W, Zimmermann P (2008) Genevestigator v3: a reference expression database for the meta-analysis of transcriptomes. Adv Bioinforma 2008, 5pp.
15. An open-source r framework for your microarray analysis (2018). http://www.aroma-project.org/. [Online; accessed Jan 2018]
16. ELVIRA Biomedical Data Set Repository (2018). http://leo.ugr.es/elvira/DBCRepository/. [Online; accessed Jan 2018]
17. Machine Learning Dataset Repository (2018). http://mldata.org/repository/data/. [Online; accessed Jan 2018]
18. The home of data science & machine learning (2018). https://www.kaggle.com/datasets. [Online; accessed Jan 2018]
19. Frank A, Asuncion A (2018). UCI machine learning repository. http://archive.ics.uci.edu/ml, 2010. [Online; accessed Jan 2018]
20. Feature Selection Datasets at Arizona State University (2018). http://featureselection.asu.edu/datasets.php. [Online; accessed Jan 2018]
21. Bioconductor, open source software for bioinformatics (2018). http://www.bioconductor.org. [Online; accessed Jan 2018]
22. Pomeroy SL, Tamayo P, Gaasenbeek M, Sturla LM, Angelo M, McLaughlin ME, Kim JYH, Goumnerova LC, Black PM, Lau C *et al* (2002) Prediction of central nervous system embryonal tumour outcome based on gene expression. Nature 415(6870):436–442
23. Shah M, Marchand M, Corbeil J (2012) Feature selection with conjunctions of decision stumps and learning from microarray data. IEEE Trans Pattern Anal Mach Intell 34 (1):174–186
24. Tian E, Zhan F, Walker R, Rasmussen E, Ma Y, Barlogie B, Shaughnessy JD Jr (2003) The role of the wnt-signaling antagonist dkk1 in the development of osteolytic lesions in multiple myeloma. N Engl J Med 349 (26):2483–2494
25. Nutt CL, Mani DR, Betensky RA, Tamayo P, Cairncross JG, Ladd C, Pohl U, Hartmann C, McLaughlin ME, Batchelor TT *et al* (2003) Gene expression-based classification of malignant gliomas correlates better with survival than histological classification. Cancer Res 63(7):1602–1607
26. Bolón-Canedo V, Seth S, Sánchez-Maroño N, Alonso-Betanzos A, Principe JC (2011) Statistical dependence measure for feature selection in microarray datasets. In: 19th European symposium on artificial neural networks-ESANN, pp 23–28

27. Bolón-Canedo V, Sánchez-Marono N, Alonso-Betanzos A, Benítez JM, Herrera F (2014) A review of microarray datasets and applied feature selection methods. Inf Sci 282:111–135
28. Bolón-Canedo V, Sechidis K, Sánchez-Marono N, Alonso-Betanzos A, Brown G (2017) Exploring the consequences of distributed feature selection in dna microarray data. In: International joint conference on neural networks
29. Ebrahimpour MK, Zare M, Eftekhari M, Aghamolaei G (2017) Occam's razor in dimension reduction: using reduced row echelon form for finding linear independent features in high dimensional microarray datasets. Eng Appl Artif Intell 62:214–221
30. Wanderley MF, Gardeux V, Natowicz R, Braga AP (2013) Ga-kde-bayes: an evolutionary wrapper method based on non-parametric density estimation applied to bioinformatics problems. In: 21st European symposium on artificial neural networks-ESANN, pp 155–160
31. Meyer PE, Schretter C, Bontempi G (2008) Information-theoretic feature selection in microarray data using variable complementarity. IEEE J Sel Top Signal Process 2 (3):261–274
32. Hedenfalk I, Duggan D, Chen Y, Radmacher M, Bittner M, Simon R, Meltzer P, Gusterson B, Esteller M, Raffeld M *et al* (2001) Gene-expression profiles in hereditary breast cancer. N Engl J Med 344 (8):539–548
33. Lee C, Leu Y (2011) A novel hybrid feature selection method for microarray data analysis. Appl Soft Comput 11(1):208–213
34. van't Veer LJ, Dai H, Van De Vijver MJ, He YD, Hart AAM, Mao M, Peterse HL, van der Kooy K, Marton MJ, Witteveen AT *et al* (2002) Gene expression profiling predicts clinical outcome of breast cancer. Nature 415(6871):530–536
35. Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A (2012) An ensemble of filters and classifiers for microarray data classification. Pattern Recogn 45(1):531–539
36. Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A (2010) On the effectiveness of discretization on gene selection of microarray data. In: The 2010 international joint conference on neural networks (IJCNN). IEEE, Piscataway, pp 18–23
37. Kumar M, Rath SK (2015) Classification of microarray using mapreduce based proximal support vector machine classifier. Knowl-Based Syst 89:584–602
38. Mohapatra P, Chakravarty S, Dash PK (2016) Microarray medical data classification using kernel ridge regression and modified cat swarm optimization based gene selection system. Swarm Evol Comput 28:144–160
39. Navarro FFG, Muñoz LAB (2009) Gene subset selection in microarray data using entropic filtering for cancer classification. Expert Syst 26(1):113–124
40. West M, Blanchette C, Dressman H, Huang E, Ishida S, Spang R, Zuzan H, Olson JA, Marks JR, Nevins JR (2001) Predicting the clinical status of human breast cancer by using gene expression profiles. Proc Natl Acad Sci 98(20):11462–11467
41. Leung Y, Hung Y (2010) A multiple-filter-multiple-wrapper approach to gene selection and microarray data classification. IEEE/ACM Trans Comput Biol Bioinform 7 (1):108–117
42. Heap G, Trynka G, Jansen R, Bruinenberg M, Swertz M, Dinesen L, Hunt K, Wijmenga C *et al* (2009) Complex nature of snp genotype effects on gene expression in primary human leucocytes. BMC Med Genomics 2(1):1
43. Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A (2014) Data classification using an ensemble of filters. Neurocomputing 135:13–20
44. Dessì N, Pes B (2015) Similarity of feature selection methods: an empirical study across data intensive classification tasks. Expert Syst Appl 42(10):4632–4642
45. Shreem SS, Abdullah S, Nazri MZA, Alzaqebah M (2012) Hybridizing ReliefF, MRMR filters and GA wrapper approaches for gene selection. J Theor Appl Inf Technol 46 (2):1034–1039
46. Yang F, Mao KZ (2011) Robust feature selection for microarray data based on multicriterion fusion. IEEE/ACM Trans Comput Biol Bioinform 8(4):1080–1092
47. Ye Y, Wu Q, Huang JZ, Ng MK, Li X (2013) Stratified sampling for feature subspace selection in random forests for high dimensional data. Pattern Recogn 46(3):769–787
48. Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D, Levine AJ (1999) Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc Natl Acad Sci 96(12):6745–6750
49. Ferreira AJ, Figueiredo MAT (2012) An unsupervised approach to feature

discretization and selection. Pattern Recogn 45(9):3048–3060

50. Lovato P, Bicego M, Cristani M, Jojic N, Perina A (2012) Feature selection using counting grids: application to microarray data. In: Structural, syntactic, and statistical pattern recognition. Springer, Berlin, pp 629–637
51. Song L, Smola A, Gretton A, Bedo J, Borgwardt K (2012) Feature selection via dependence maximization. J Mach Learn Res 98888:1393–1434
52. Maldonado S, Weber R, Basak J (2011) Simultaneous feature selection and classification using kernel-penalized support vector machines. Inf Sci 181(1):115–128
53. Abeel T, Helleputte T, Van de Peer Y, Dupont P, Saeys Y (2010) Robust biomarker identification for cancer diagnosis with ensemble feature selection methods. Bioinformatics 26(3):392–398
54. Mundra PA, Rajapakse JC (2010) SVM-RFE with mRMR filter for gene selection. IEEE Trans NanoBiosci 9(1):31–37
55. Nguyen T, Khosravi A, Creighton D, Nahavandi S (2015) Hidden Markov models for cancer classification using gene expression profiles. Inf Sci 316:293–307
56. Wang J, Wu L, Kong J, Li Y, Zhang B (2013) Maximum weight and minimum redundancy: a novel framework for feature subset selection. Pattern Recogn 46(6):1616–1627
57. Song Q, Ni J, Wang G (2013) A fast clustering-based feature subset selection algorithm for high-dimensional data. IEEE Trans Knowl Data Eng 25(1):1–14
58. Canul-Reich J, Hall LO, Goldgof DB, Korecki JN, Eschrich S (2012) Iterative feature perturbation as a gene selector for microarray data. Int J Pattern Recogn Artif Intell 26 (05):1260003
59. Moradkhani M, Amiri A, Javaherian M, Safari H (2015) A hybrid algorithm for feature subset selection in high-dimensional datasets using FICA and IWSSr algorithm. Appl Soft Comput 35:123–135
60. Noble CL, Abbas AR, Cornelius J, Lees CW, Ho G, Toy K, Modrusan Z, Pal N, Zhong F, Chalasani S *et al* (2008) Regional variation in gene expression in the healthy colon is dysregulated in ulcerative colitis. Gut 57 (10):1398–1405
61. Shipp MA, Ross KN, Tamayo P, Weng AP, Kutok JL, Aguiar RCT, Gaasenbeek M, Angelo M, Reich M, Pinkus GS *et al* (2002) Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. Nat Med 8 (1):68–74
62. Chuang L, Yang C, Wu K, Yang C (2011) A hybrid feature selection method for dna microarray data. Comput Biol Med 41 (4):228–237
63. Alizadeh AA, Eisen MB, Davis RE, Ma C, Lossos IS, Rosenwald A, Boldrick JC, Sabet H, Tran T, Yu X *et al* (2000) Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. Nature 403 (6769):503–511
64. Freije WA, Castro-Vargas FE, Fang Z, Horvath S, Cloughesy T, Liau LM, Mischel PS, Nelson SF (2004) Gene expression profiling of gliomas strongly predicts survival. Cancer Res 64(18):6503–6510
65. Nie F, Huang H, Cai X, Ding C (2010) Efficient and robust feature selection via joint l2, 1-norms minimization. Adv Neural Inf Process Syst 23:1813–1821
66. Guangtao W, Qinbao S, Baowen X, Yuming Z (2013) Selecting feature subset for high dimensional data via the propositional foil rules. Pattern Recogn 46(1):199–214
67. Kang S, Song J (2017) Robust gene selection methods using weighting schemes for microarray data analysis. BMC Bioinformatics 18 (1):389
68. Garber ME, Troyanskaya OG, Schluens K, Petersen S, Thaesler Z, Pacyna-Gengelbach-M, Van De Rijn M, Rosen GD, Perou CM, Whyte RI *et al* (2001) Diversity of gene expression in adenocarcinoma of the lung. Proc Natl Acad Sci 98(24):13784–13789
69. Gordon GJ, Jensen RV, Hsiao L, Gullans SR, Blumenstock JE, Ramaswamy S, Richards WG, Sugarbaker DJ, Bueno R (2002) Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. Cancer Res 62(17):4963–4967
70. Zhou P, Hu X, Li P, Wu X (2017) Online feature selection for high-dimensional class-imbalanced data. Knowl-Based Syst 136:187–199
71. Shedden K, Taylor JMG, Enkemann SA, Tsao M, Yeatman TJ, Gerald WL, Eschrich S, Jurisica I, Giordano TJ, Misek DE *et al* (2008) Gene expression–based survival prediction in lung adenocarcinoma: a multi-site, blinded validation study. Nat Med 14(8):822–827

72. Eschrich S, Yang I, Bloom G, Kwong KY, Boulware D, Cantor A, Coppola D, Kruhøffer M, Aaltonen L, Orntoft TF *et al* (2005) Molecular staging for survival prediction of colorectal cancer patients. J Clin Oncol 23(15):3526–3535
73. Petricoin EF, Ardekani AM, Hitt BA, Levine PJ, Fusaro VA, Steinberg SM, Mills GB, Simone C, Fishman DA, Kohn EC *et al* (2002) Use of proteomic patterns in serum to identify ovarian cancer. Lancet 359 (9306):572–577
74. Singh D, Febbo PG, Ross K, Jackson DG, Manola J, Ladd C, Tamayo P, Renshaw AA, D'Amico AV, Richie JP *et al* (2002) Gene expression correlates of clinical prostate cancer behavior. Cancer Cell 1(2):203–209
75. Sharma A, Imoto S, Miyano S (2012) A top-r feature selection algorithm for microarray gene expression data. IEEE/ACM Trans Comput Biol Bioinform 9(3):754–764
76. Spira A, Beane JE, Shah V, Steiling K, Liu G, Schembri F, Gilman S, Dumas Y, Calner P, Sebastiani P *et al* (2007) Airway epithelial gene expression in the diagnostic evaluation of smokers with suspect lung cancer. Nat Med 13(3):361–366
77. Staunton JE, Slonim DK, Coller HA, Tamayo P, Angelo MJ, Park J, Scherf U, Lee JK, Reinhold WO, Weinstein JN *et al* (2001) Chemosensitivity prediction by transcriptional profiling. Proc Natl Acad Sci 98 (19):10787–10792
78. Liu Z, Tang D, Cai Y, Wang R, Chen F (2017) A hybrid method based on ensemble welm for handling multi class imbalance in cancer microarray data. Neurocomputing 266:641–650
79. Su AI, Welsh JB, Sapinoso LM, Kern SG, Dimitrov P, Lapp H, Schultz PG, Powell SM, Moskaluk CA, Frierson HF Jr *et al* (2001) Molecular classification of human carcinomas by use of gene expression signatures. Cancer Res 61(20):7388–7393
80. Liu K-H, Zeng Z-H, Ng VTY (2016) A hierarchical ensemble of ECOC for cancer classification based on multi-class microarray data. Inf Sci 349:102–118
81. Ramaswamy S, Tamayo P, Rifkin R, Mukherjee S, Yeang C, Angelo M, Ladd C, Reich M, Latulippe E, Mesirov JP *et al* (2001) Multiclass cancer diagnosis using tumor gene expression signatures. Proc Natl Acad Sci 98 (26):15149–15154
82. Lan L, Vucetic S (2011) Improving accuracy of microarray classification by a simple multi-task feature selection filter. Int J Data Min Bioinform 5(2):189–208
83. Morán-Fernández L, Bolón-Canedo V, Alonso-Betanzos A (2017) On the use of different base classifiers in multiclass problems. Prog Artif Intell 1–9. https://doi.org/10.1007/s13748-017-0126-4
84. Haslinger C, Schweifer N, Stilgenbauer S, Döhner H, Lichter P, Kraut N, Stratowa C, Abseher R (2004) Microarray gene expression profiling of B-cell chronic lymphocytic leukemia subgroups defined by genomic aberrations and VH mutation status. J Clin Oncol 22(19):3937–3949
85. Sun L, Hui A, Su Q, Vortmeyer A, Kotliarov Y, Pastorino S, Passaniti A, Menon J, Walling J, Bailey R *et al* (2006) Neuronal and glioma-derived stem cell factor induces angiogenesis within the brain. Cancer Cell 9(4):287–300
86. Anaissi A, Kennedy PJ, Goyal M (2011) Feature selection of imbalanced gene expression microarray data. In: 2011 12th ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing (SNPD). IEEE, Piscataway, pp 73–78
87. Armstrong SA, Staunton JE, Silverman LB, Pieters R, den Boer ML, Minden MD, Sallan SE, Lander ES, Golub TR, Korsmeyer SJ *et al* (2002) Mll translocations specify a distinct gene expression profile that distinguishes a unique leukemia. Nat Genet 30(1):41–47
88. Student S, Fujarewicz K (2012) Stable feature selection and classification algorithms for multiclass microarray data. Biol Direct 7 (1):33
89. Liu K-H, Tong M, Xie S-T, Ng VTY (2015) Genetic programming based ensemble system for microarray data classification. Comput Math Methods Med 2015, 11pp.
90. Bhattacharjee A, Richards WG, Staunton J, Li C, Monti S, Vasa P, Ladd C, Beheshti J, Bueno R, Gillette M *et al* (2001) Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses. Proc Natl Acad Sci 98 (24):13790–13795
91. Stienstra R, Saudale F, Duval C, Keshtkar S, Groener JEM, van Rooijen N, Staels B, Kersten S, Müller M (2010) Kupffer cells promote hepatic steatosis via interleukin-1beta-dependent suppression of peroxisome proliferator-activated receptor alpha activity. Hepatology 51(2):511–522

92. Khan J, Wei JS, Ringner M, Saal LH, Ladanyi M, Westermann F, Berthold F, Schwab M, Antonescu CR, Peterson C *et al* (2001) Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. Nat Med 7 (6):673–679
93. Brown MPS, Grundy WN, Lin D, Cristianini N, Sugnet CW, Furey TS, Ares M, Haussler D (2000) Knowledge-based analysis of microarray gene expression data by using support vector machines. Proc Natl Acad Sci 97(1):262–267
94. Dougherty ER (2001) Small sample issues for microarray-based classification. Comp Funct Genomics 2(1):28–34
95. Yang H, Churchill G (2007) Estimating p-values in small microarray experiments. Bioinformatics 23(1):38–43
96. Storey JD, Tibshirani R, Garret ES, Irizarry RA, Zeger SL (2003) SAM thresholding and false discovery rates for detecting differential gene expression in DNA microarrays. Springer, New York
97. Xie Y, Pan W, Khodursky AB (2005) A note on using permutation-based false discovery rate estimates to compare different analysis methods for microarray data. Bioinformatics 21(23):4280–4288
98. Murie C, Woody O, Lee AY (2009) Comparison of small n statistical tests of differential expression applied to microarrays. BMC Bioinformatics 10:45
99. Paul J, Chiu D, Golovan S, Husain M, Hakimov H (2008) Analysis of extremely small sample microarrays using multi-source data 1
100. Nikulin V (2014) On a solution for the high-dimensionality-small-sample-size regression problem with several different microarrays. Int J Data Min Bioinform 9(3):221–234
101. Allison DB, Gadbury GL, Heo M, Fernández JR, Lee C-K, Prolla TA, Weindruch R (2002) A mixture model approach for the analysis of microarray gene expression data. Comput Stat Data Anal 39(1):1–20
102. Phan JH, Moffitt RA, Barrett AB, Wang MD (2008) Improving microarray sample size using bootstrap data combination. In: Proceedings conf. IEEE engineering in medicine and biology society. IEEE, Piscataway, pp 5660–5663
103. Braga-Neto U (2007) Fads and fallacies in the name of small-sample microarray classification-a highlight of misunderstanding and erroneous usage in the applications of genomic signal processing. IEEE Signal Process Mag 24(1):91–99
104. Michiels S, Koscielny S, Hill C (2005) Prediction of cancer outcome with microarrays: a multiple random validation strategy. Lancet 365(9458):488–492
105. Braga-Neto UM, Dogherty ER (2004) Is cross-validation valid for small-sample microarray classification? Bioinformatics 20 (3):374–380
106. Hanczar B, Jianping H, Sima C, Weinstein J, Bittner M, Dougherty ER (2010) Small-sample precision of ROC-related estimates. Bioinformatics 26(6):822–830
107. Laber EB, Murphy SA (2008) Small sample inference for generalization error in classification using the cud bound. In: Proc. of the conference on uncertainty in artificial intelligence, pp 357–365
108. He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284
109. Sun Y, Wong AKC, Kamel MS (2009) Classification of imbalanced data: a review. Int J Pattern Recogn Artif Intell 23(04):687–719
110. López V, Fernández A, García S, Palade V, Herrera F (2013) An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics. Inf Sci 250(0):113–141
111. Lusa L *et al* (2010) Class prediction for high-dimensional class-imbalanced data. BMC Bioinformatics 11(1):523
112. Galar M, Fernández A, Barrenechea E, Bustince H, Herrera F (2012) A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Trans Syst Man Cybern Part C Appl Rev 42(4):463–484
113. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357
114. Blagus R, Lusa L (2012) Evaluation of smote for high-dimensional class-imbalanced microarray data. In: 2012 11th international conference on machine learning and applications (ICMLA), vol 2. IEEE, Piscataway, pp 89–94
115. Morán-Fernández L, Bolón-Canedo V, Alonso-Betanzos A (2016) Data complexity measures for analyzing the effect of smote over microarrays. In: European symposium on artificial neural networks, computational intelligence and machine learning
116. Galar M, Fernández A, Barrenechea E, Herrera F (2013) Eusboost: enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. Pattern Recogn 46 (12):3460–3471

117. Tax DMJ, Duin RPW (2004) Support vector data description. Mach Learn 54(1):45–66
118. Maldonado S, Weber R, Famili F (2014) Feature selection for high-dimensional class-imbalanced data sets using support vector machines. Inf Sci 286:228–246
119. Ho TK, Basu M (2002) Complexity measures of supervised classification problems. IEEE Trans Pattern Anal Mach Intell 24 (3):289–300
120. Lorena AC, Costa IG, Spolaôr N, de Souto MCP (2012) Analysis of complexity indices for classification problems: cancer gene expression data. Neurocomputing 75 (1):33–42
121. Okun O, Priisalu H (2009) Dataset complexity in gene expression based cancer classification using ensembles of k-nearest neighbors. Artif Intell Med 45(2):151–162
122. Bolón-Canedo V, Moran-Fernandez L, Alonso-Betanzos A (2015) An insight on complexity measures and classification in microarray data. In: 2015 International joint conference on neural networks (IJCNN). IEEE, Piscataway, pp 42–49
123. Morán-Fernández L, Bolón-Canedo V, Alonso-Betanzos A (2017) Can classification performance be predicted by complexity measures? A study using microarray data. Knowl Inf Syst 51(3):1067–1090
124. Moreno-Torres JG, Raeder T, Alaiz-Rodríguez R, Chawla NV, Herrera F (2012) A unifying view on dataset shift in classification. Pattern Recogn 45(1):521–530
125. Moreno-Torres JG, Sáez JA, Herrera F (2012) Study on the impact of partition-induced dataset shift on k-fold cross-validation. IEEE Trans Neural Netw Learn Syst 23 (8):1304–1312
126. Barnett V, Lewis T (1994) Outliers in statistical data, vol 3. Wiley, New York
127. Kadota K, Tominaga D, Akiyama Y, Takahashi K (2003) Detecting outlying samples in microarray data: a critical assessment of the effect of outliers on sample classification. Chem-Bio Inf 3(1):30–45
128. Gonzalez-Navarro FF (2011) Feature selection in cancer research: microarray gene expression and in vivo 1H-MRS domains. PhD thesis, Technical University of Catalonia

# Chapter 5

# Statistical Analysis of Microarray Data

Ricardo Gonzalo Sanz and Alex Sánchez-Pla

## Abstract

Microarray data analysis has been one of the most important hits in the interaction between statistics and bioinformatics in the last two decades. The analysis of microarray data can be done in different ways using different tools. In this chapter a typical workflow for analyzing microarray data using R and Bioconductor packages is presented. The workflow starts with the raw data—binary files obtained from the hybridization process—and goes through a series of steps: Reading raw data, Quality Check, Normalization, Filtering, Selection of differentially expressed genes, Comparison of selected lists, and Analysis of Biological Significance. The implementation of each step in R is described through a use case that goes from raw data until the analysis of biological significance. Data and code for the analysis are provided in a github repository.

**Key words** Microarrays, Bioconductor, R, Differential expression

## 1 Introduction

Microarray data analysis is one of the clearest cases where interaction between bioinformatics and statistics has been highly beneficial for both disciplines. Efron [1] even calls the twenty-first century as the century of microarrays.

What is generically described as "microarray data analysis" is a process that starts with the design of the experiment intended to answer with one or more biological questions and ends with a tentative answer for these questions. Statistics is involved at every step of this process, for preparing, transforming visualizing or analyzing data. And, of course, every step can be done in different way that use either classical statistics or new methods developed ad hoc for these often high-dimensional problems. The detailed description of these steps is out of the scope of this chapter and the reader is assumed to be familiar with them. It is assumed that the reader is already familiar with microarrays such as they are introduced in [2] and also with the general ideas of microarray data analysis such as can be found in [3]. In any case, for the sake

Verónica Bolón-Canedo and Amparo Alonso-Betanzos (eds.), *Microarray Bioinformatics*, Methods in Molecular Biology, vol. 1986, https://doi.org/10.1007/978-1-4939-9442-7_5, 

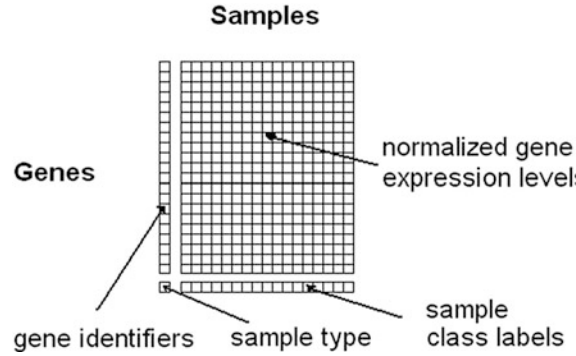


**Fig. 1** A simplified view of a gene expression matrix

of completeness basic ideas will be briefly introduced and citations provided the first time they are discussed.

For our objectives we can assume that a microarray dataset is a matrix of continuous values that represent the expressions of a set of genes (one gene per row), in a variety of conditions or samples (one sample per column). *See* Fig. 1 for an example.

Note that we have described the row contents as "genes." Strictly speaking, depending on the type of array, each row may correspond to one distinct, but related, entity, a "probeset" or a "transcript."

- A transcript describes how the gene has been transcribed into messenger RNA. If transcription was unique there would be a single transcript per gene. However, due to the phenomenon of alternative splicing, [4], there may be different transcriptions of the same gene (the associated proteins are called "isoforms"). Note that there may be multiple transcripts per gene.
- A probeset is, as indicated by its name, a set of "probes," which are designed to map different fragments of a given gene. Altogether it is expected that each set of probes, or probeset, uniquely characterizes one gene. However, given that this characterization is not always possible it may be convenient to have more than one probeset per gene. That is, although it is common to exchange the terms "probeset" and "gene," it is important to be aware that there may be several probesets per each gene.

In practice, given that either probesets or transcripts map to genes, it is common to describe the array rows as "genes."

Our main goal is to describe a workflow, a series of ordered steps that takes us from the raw data, the digitized images as produced by the hybridization system, to one or more lists of genes that can be used to help answering a certain biological question. This can be done in distinct ways. What we present here is an approach that has become very popular over the last decades

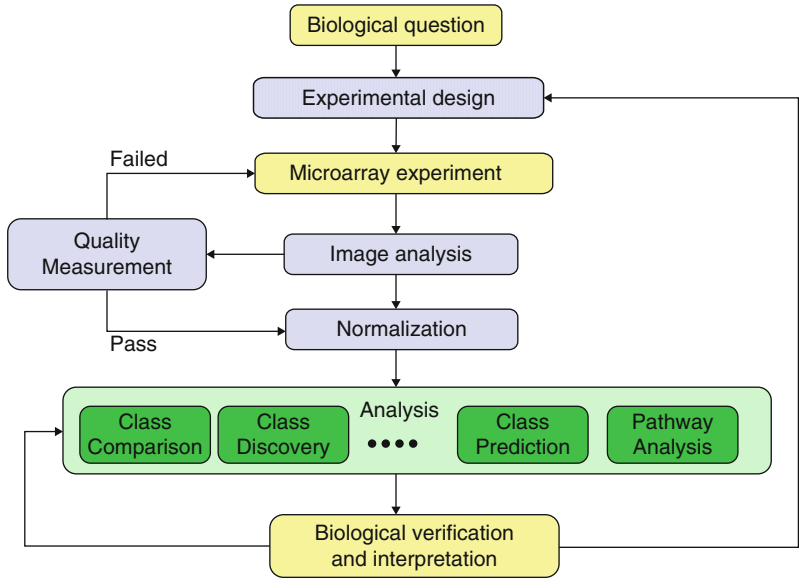


**Fig. 2** Microarray data analysis process

based on analyzing the data from the images to the lists of genes, using the R Statistical language and some of the packages developed specifically for this in the Bioconductor project.

A summary of the process can be found in Fig. 2.

## 2 Materials

In this section we list all the materials needed to perform a microarray data analysis.

### *2.1 Software*

First of all, it is needed to install the software to perform all the required calculations.

There are many options available [5], but one of the most common approaches is to use the **R statistical software**. R can be downloaded from its web page (https://cran.r-project.org/index.html) and installed following the instructions described there. The microarray analysis presented in this chapter has been performed with the latest version of R which, at the moment of writing, was 3.4.4.

R is a console-based software. Its use can be facilitated with an additional interface called "R-Studio." It can be downloaded and installed following the instructions listed in its web page: https://www.rstudio.com/. Although its use is not compulsory for reproducing the analyses in this chapter it is highly recommended to work with R using this interface.

When working with R it is often required to use some functions not available in the basic installation. This can be done by installing additional libraries, also called "packages." developed by the scientific community. Most packages used for the analysis of high throughput genomic data are part of the Bioconductor project which started with a few packages in 2002 and has now more than one thousand (https://www.bioconductor.org/). Indeed, Bioconductor has become the state-of-the-art way to analyze microarray and other omics data and it has grown from hardly a dozen packages in 2002 to the current number of more than one thousand. The analysis presented have been performed using Bioconductor version 3.6.

R and Bioconductor are open source free software. They have many advantages, but they may sometimes be a problem, especially when new functionalities are not compatible with previous versions (*see* **Note 1**).

Table 1 shows the packages needed for the analysis presented in this chapter. The table contains the name, the source, and a short description of all the packages that have to be install to run the current case study. In the following section we will show the code necessary to install all of them.

### 2.2 Data

This protocol is applied on a dataset from a published study [20]. The data had been uploaded into the Gene Expression Omnibus (GEO) database, an international public repository that archives and freely distributes high-throughput gene expression and other functional genomics data sets [21]. The dataset selected is identified with the accession number: **GSE100924**.

The study that generated the data investigated the function of gene ZBTB7B (http://www.genecards.org/cgi-bin/carddisp.pl?gene=ZBTB7B). This gene activates the thermogenic gene program during brown and beige adipocyte differentiation regulating brown fat gene expression at ambient room temperature and following cold exposure. The experiment compared 10-week-old mice with the gene deactivated ("KO" or knockout) or not ("WT" or Wild type) at two different temperatures, ambient room temperature (RT, 22 °C) or following cold exposure (COLD, 4 °C) for 4 h. That is, it was a 2 × 2 factorial design (genotype and temperature) with two levels each (wild type and knockout, for genotype, and room temperature and cold, for temperature). The sample size of the experiment is 12 samples, three replicates of each group.

The microarrays used for this experiment are of type Mouse Gene 2.1 from Affymetrix, now Thermofisher, one of the most popular vendors of microarray technology.

**Table 1**
**List of R/Bioconductor packages used in this chapter**

| Package.name | Description | References |
|---|---|---|
| Oligo | Oligo is a Bioconductor package for preprocessing oligonucleotide microarrays. We use this package to read the CEL files and normalize them | [6] |
| pd.mogene.2.1.st | It is an annotation package for the array used in this experiment | [7] |
| Biobase | This package contains standardized data structures to represent genomic data that are used by other R packages | [8] |
| arrayQualityMetrics | This package generates microarray quality metrics reports for microarray data | [9] |
| gplots | Various R programming tools for plotting data | [10] |
| ggplot2 | This package is a plotting system for R. We will use it to build nice graphs | [11] |
| ggrepel | This package is used to avoid overlapping of labels in plots | [12] |
| pvca | This package contains the functions necessary to assess the batch effect sources involved in a microarray experiment | [13] |
| limma | Limma is an R package for the analysis of gene expression microarray data especially the use of linear models for analyzing designed experiments and the assessment of differential expression | [14] |
| genefilter | The genefilter package can be used to filter (select) genes from a microarray dataset according to a variety of different filtering mechanisms | [15] |
| annotate | An R package for managing annotations, that is, for providing functions to extract information from annotation packages. These are essentially databases containing information about genes and related features | [16] |
| org.Mm.eg.db | Annotation package for mouse. Contains multiple SQL tables relating gene information from mouse with different databases. The main key for all tables is the Entrez Gene identifier a unique integer number for every gene in every organism. There are multiple organism-centric annotation packages named as: org.XX.eg.db. Each package provides information on organism XX indexed by the Entrez Gene identifiers (eg) of this organism | [17] |
| mogene21sttranscriptcluster.db | Platform-centric annotation package. Instead of providing information using the Entrez Gene identifiers these databases use the specific c identifiers of (at the probeset or transcript level) of a given type of array as indexes to link to other databases | [18] |
| ReactomePA | An R/Bioconductor package providing enrichment analyses including hypergeometric test and gene set enrichment analyses | [19] |

## 3 Methods

### 3.1 Environment Preparation

In a microarray data analysis project, data analyst will have to manage a lot of files, including the files with the raw data (.CEL files) and the files generated during the analysis of them. For this reason, it is very advisable to define some folders before beginning with the analysis to try to not get lost, if all the files go to the same folder. We strongly recommend that user creates the following folders:

- A main folder that will be the "working directory" called, for example, "MicroarraysAnalysis."
- A folder called, for example, **data** located within the working directory: Here we will save all the *.CEL* files and the *targets* file with information on the covariates, described in next section.
- A folder called, for example, **results** located within the working directory: Here we will send all the results obtained in the microarray analysis.

The following commands create the desired folders. This can be made from within R as described, or using a visual file browser such as Windows File Explorer or any other (in that case you can omit this step):

```
> dir.create("data")
> dir.create("results")
```

The code for running this analysis, that can be easily adapted to run similar studies can be downloaded from a github repository specifically devoted to this chapter. In the following sections it is assumed that such code has been downloaded and copied into the working directory. The url for the repository is: https://github.com/alexsanchezpla/StatisticalAnalysisOfMicroarrayData.

Once it is saved, open R-Studio, open the file with the R code, and in R-Studio go to the menu option in the top "Session -> Set working directory -> To source file location." This action will set the folder we have set as "main" folder as our working directory.

### 3.2 Prepare the Data for the Analysis

The data for the analysis will be provided as two types of files, the "CEL" "files and the "targets" file.

CEL files are the files with the "raw data" originated after microarray scanning and preprocessing with Affymetrix software. These files need to be saved into the **data** folder. Usually one expects to have a .CEL file for each sample in the experiment.

Another file needed for the analysis is the *targets* file, which contains the information on groups and covariates. That is, this file relates the name of each .CEL file with their condition in the

experiment. We can use the targets to retain all the information valuable for the analysis like other covariables.

Although the targets file need not have any fixed names it is practical to use its column names to create labels that will be used later. For example:

- Column called *FileName*: It may contain the exact name of the CEL files in the data folder.
- Column called *Group*: It may summarize the conditions in the experiment for that sample.
- Column called *ShortName*: It may be used to store a short label of the sample useful for some plots.
- There may be other columns to store covariables in the study such as sex and age.

For this analysis, targets file has been saved in .csv format, separated by semicolon, although any other format that works for delimited text files might have been used (*see* **Note 2**). Table 2 shows the contents of the targets file used in this analysis.

```
> targets <- read.csv2("./data/targets.csv", header = TRUE, sep = ";")
> knitr::kable(
+   targets, booktabs = TRUE,
+   caption = 'Content of the targets file used for the current
    analysis')
```

**Table 2**
**Content of the targets file used for the current analysis**

| FileName | Group | Genotype | Temperature | ShortName |
|---|---|---|---|---|
| GSM2696488_WT_RT_1.CEL | WT.RT | WT | RT | WT.RT.1 |
| GSM2696489_WT_RT_2.CEL | WT.RT | WT | RT | WT.RT.2 |
| GSM2696490_WT_RT_3.CEL | WT.RT | WT | RT | WT.RT.3 |
| GSM2696491_KO_RT_1.CEL | KO.RT | KO | RT | KO.RT.1 |
| GSM2696492_KO_RT_2.CEL | KO.RT | KO | RT | KO.RT.2 |
| GSM2696493_KO_RT_3.CEL | KO.RT | KO | RT | KO.RT.3 |
| GSM2696494_WT_Cold_1.CEL | WT.COLD | WT | COLD | WT.COLD.1 |
| GSM2696495_WT_Cold_2.CEL | WT.COLD | WT | COLD | WT.COLD.2 |
| GSM2696496_WT_Cold_3.CEL | WT.COLD | WT | COLD | WT.COLD.3 |
| GSM2696497_KO_Cold_1.CEL | KO.COLD | KO | COLD | KO.COLD.1 |
| GSM2696498_KO_Cold_2.CEL | KO.COLD | KO | COLD | KO.COLD.2 |
| GSM2696499_KO_Cold_3.CEL | KO.COLD | KO | COLD | KO.COLD.3 |

### 3.3 Packages Installation in R

Packages not available in the basic R installation need to be installed before the analysis can be done (*see* **Note 3**).

As commented in Subheading 2, packages needed to do the study, may be downloaded from distinct repositories. The most common ones will be CRAN for standard packages or Bioconductor for Bioconductor packages.

Standard R packages can be downloaded and installed from default repositories with the install.packages function. Bioconductor packages can be downloaded and installed with the function biocLite() which at its time can be loaded into R with the instruction source ("https://bioconductor.org/biocLite.R").

The code below will download and install the packages needed for the analysis. Note that this code must be executed only once. Subsequent executions of the analysis do not need to reinstall the packages:

```
> install.packages("colorspace")
> install.packages("gplots")
> install.packages("ggplot2")
> install.packages("ggrepel")
> source("http://bioconductor.org/biocLite.R")
> biocLite("oligo")
> biocLite("pd.mogene.2.1.st")
> biocLite("arrayQualityMetrics")
> biocLite("limma")
> biocLite("genefilter")
> biocLite("pvca")
> biocLite("mogene21sttranscriptcluster.db")
> biocLite("annotate")
> biocLite("org.Mm.eg.db")
> biocLite("ReactomePA")
```

### 3.4 Read the CEL Files

Next step is to read the raw data (CEL files) and to store in a variable (in this case we have called it **rawData**). First, we have to load the package *oligo* with the function *library*. In this package are coded the functions to read the CEL files. Take care to put the correct folder where the CEL files are saved when executing *list.celfiles* function.

```
> require(oligo)
> celFiles <- list.celfiles("./data", full.names = TRUE)
> require(Biobase)
> my.targets <-read.AnnotatedDataFrame(file.path("./data","targets.csv"),
+                                      header = TRUE, row.names = 1,
+                                      sep=";")
> rawData <- read.celfiles(celFiles, phenoData = my.targets)
```

Note that we have read another time the targets file, but now using another specific function: *read.AnnotatedDataFrame*, and

stored in a new variable called *my.targets*. We have done that to associate the information stored in the CEL files with the targets file in on single variable with the last code's line. This object is called *ExpressionSet* and is designed to combine several different sources of information into a single convenient structure. We could store in this object all the information available about the experiment performed (protocol used, experiment data, microarray type, ...). Moreover, it allows us to change the long name of the samples, for the short and more comprehensive label previously coded in *ShortName* column of the *targets*.

```
> colnames(rawData) <-rownames(pData(rawData)) <-
my.targets@data$ShortName
```

### 3.5 Quality Control of Raw Data

Once the raw data is loaded it is the moment to check if the data have enough quality for normalization. This step is very important since bad quality data could introduce a lot of noise in the analysis, that normalization process could not solve. ArrayQualityMetrics package performs different quality approaches, like boxplot of the intensity of the data and Principal Component Analysis (PCA) among others. If one array is above a certain threshold defined in the function it is marked with an asterisk as an outlier. When a certain array is marked three times it should be revised carefully; perhaps this sample will have to be rejected to improve the overall quality of the experiment. The first step is to load the library to gain access to the function. Be careful again to specify correctly the destination folder of the results:

```
> require(arrayQualityMetrics)
> arrayQualityMetrics(rawData, outdir = file.path("./results",
"QCDir.Raw"), force=TRUE)
```

We have to check the results of the quality analysis in a recently created QCDir.Raw folder inside the results folder previously created. Inside this folder we have to look for a file called *index.html*, which opens a web page from where we will be able to access a summary of the analysis performed. The image in Fig. 3 shows the header of this file which contains a table with three columns indicating some quality criteria that should be verified by "good quality" arrays. In this example three samples have been marked once. Usually if there is only one mark it means that potential problems are small so we can decide to keep all the arrays in the analysis.

A more comprehensive principal component analysis can be obtained using a function specifically design for that. The code for this function is shown in the next code chunk.

| | array | sampleNames | *1 | *2 | *3 | Group | Genotype | Temperature | ShortName |
|---|---|---|---|---|---|---|---|---|---|
| ☐ | 1 | WT.RT.1 | | | | WT.RT | WT | RT | WT.RT.1 |
| ☐ | 2 | WT.RT.2 | | | | WT.RT | WT | RT | WT.RT.2 |
| ☐ | 3 | WT.RT.3 | | | | WT.RT | WT | RT | WT.RT.3 |
| ☐ | 4 | KO.RT.1 | | | | KO.RT | KO | RT | KO.RT.1 |
| ☐ | 5 | KO.RT.2 | | | | KO.RT | KO | RT | KO.RT.2 |
| ☐ | 6 | KO.RT.3 | | | | KO.RT | KO | RT | KO.RT.3 |
| ☐ | 7 | WT.COLD.1 | | | | WT.COLD | WT | COLD | WT.COLD.1 |
| ☐ | 8 | WT.COLD.2 | | | x | WT.COLD | WT | COLD | WT.COLD.2 |
| ☐ | 9 | WT.COLD.3 | | | x | WT.COLD | WT | COLD | WT.COLD.3 |
| ☐ | 10 | KO.COLD.1 | | | | KO.COLD | KO | COLD | KO.COLD.1 |
| ☐ | 11 | KO.COLD.2 | | x | | KO.COLD | KO | COLD | KO.COLD.2 |
| ☐ | 12 | KO.COLD.3 | | | | KO.COLD | KO | COLD | KO.COLD.3 |

**Fig. 3** Aspect of the summary table, in the index.html file, produced by the arrayQualityMetrics package on the raw data

```
> require(ggplot2)
> require(ggrepel)
> plotPCA3 <- function (datos, labels, factor, title, scale,colores, size
= 1.5, glineas = 0.25) {
+   data <- prcomp(t(datos),scale=scale)
+   # plot adjustments
+   dataDf <- data.frame(data$x)
+   Group <- factor
+   loads <- round(data$sdev^2/sum(data$sdev^2)*100,1)
+   # main plot
+   p1 <- ggplot(dataDf,aes(x=PC1, y=PC2)) +
+     theme_classic() +
+     geom_hline(yintercept = 0, color = "gray70") +
+     geom_vline(xintercept = 0, color = "gray70") +
+     geom_point(aes(color = Group), alpha = 0.55, size = 3) +
+     coord_cartesian(xlim = c(min(data$x[,1])-5,max(data$x[,1])+5)) +
+     scale_fill_discrete(name = "Group")
+  # avoiding labels superposition
+   p1 + geom_text_repel(aes(y = PC2 + 0.25, label = labels),segment.size =
0.25, size = size) +
+     labs(x =
c(paste("PC1",loads[1],"%")),y=c(paste("PC2",loads[2],"%"))) +
+     ggtitle(paste("Principal Component Analysis for: ",title,sep=" "))+
+     theme(plot.title = element_text(hjust = 0.5)) +
+     scale_color_manual(values=colores)
+   }
```

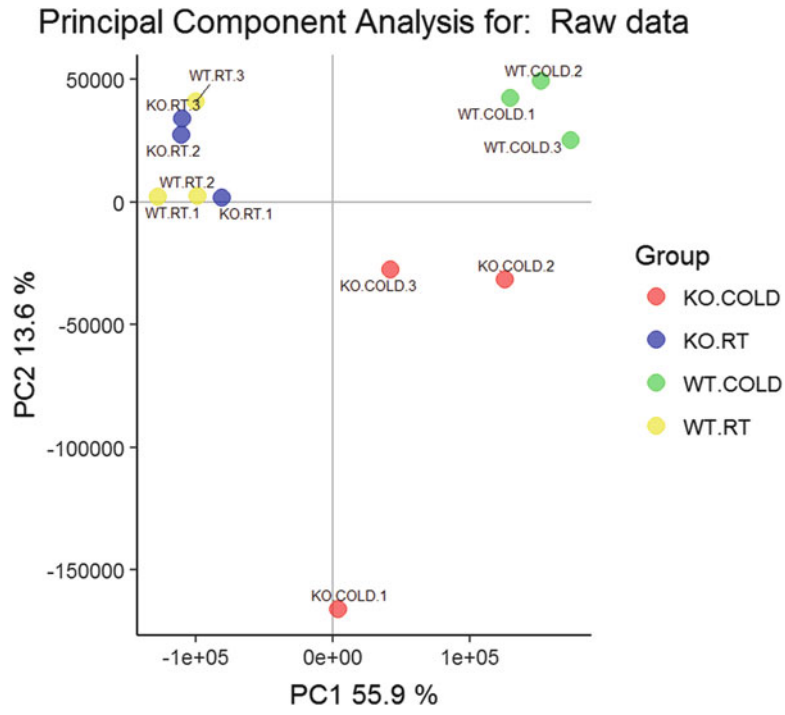


**Fig. 4** Visualization of the two first principal components for raw data

Figure 4 shows the scatterplot of the first two principal components performed on the raw data.

```
> plotPCA3(exprs(rawData), labels = targets$ShortName, factor
= targets$Group,
+          title="Raw data", scale = FALSE, size = 3,
+          colores = c("red", "blue", "green", "yellow"))
```

Note that we have defined in the function some parameters to facilitate the visualization.

- The label of the samples; remember that it is coded in the *ShortName* column of the targets.
- The characteristic to color the samples, coded in the *Group* column in targets.
- The colors of each group.

If necessary, it is easy to save the plots to a *tiff* file with the following code:

```
> tiff("figures/PCA_RawData.tiff", res = 300, width = 4, height
= 4, units = 'in')
> plotPCA3(exprs(rawData), labels = targets$ShortName, factor
= targets$Group,
+          title="Raw data", scale = FALSE, size = 2,
+          colores = c("red", "blue", "green", "yellow"))
> dev.off()
```

First component of the PCA accounts for 55.9% of the total variability of the samples, and as we can observe in the plot, this variability is mainly contributed by the *temperature* condition since samples incubated to 4° are on the right and samples incubated at room temperature are on the left.

In the same way, we can easily visualize the intensity distribution of the arrays using boxplots. Figure 5 shows a multiple boxplot depicting the distribution of the intensities along all samples.

```
> boxplot(rawData, cex.axis=0.5, las=2, which="all",
+         col = c(rep("red", 3), rep("blue", 3), rep("green", 3),
rep("yellow", 3)),
+          main="Distribution of raw intensity values")
```

A light variation of intensity among arrays is observed, but this is the expected for raw data.

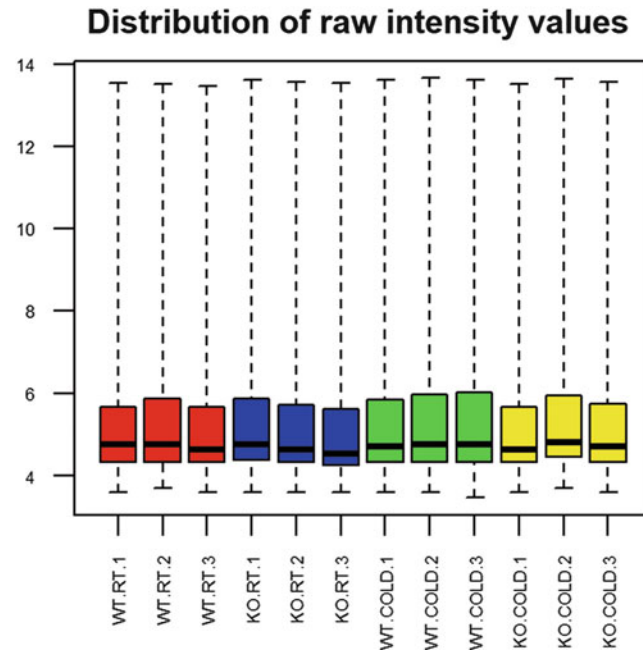


**Fig. 5** Boxplot for arrays intensities (Raw Data)

### 3.6 Data Normalization

Before beginning with differential expression analysis, it is necessary to make the arrays comparable among them and try to reduce, and if it is possible to eliminate, all the variability in the samples not owing to biological reasons. Normalization process tries to assure that the intensity differences present in the array reflect the differential expression of the genes rather than artificial biases due to technical issues. Normalization process consists of three discrete steps: background correction, normalization, and summarization. Most commonly used method for array normalization is Robust Multichip Analysis [22]:

```
> eset_rma <- rma(rawData)
Background correcting
Normalizing
Calculating Expression
```

### 3.7 Quality Control of Normalized Data

After performing normalization it is interesting to perform again a quality control to check how the data looks. In the same way than before (look, we have changed *rawData* object to *eset_rma*).

```
> arrayQualityMetrics(eset_rma, outdir = file.path("./results", "QCDir.Norm"), force=TRUE)
```

Figure 6 shows the same summary as before, but performed on normalized data.

Figure 7 shows the scatterplot of the first two principal components performed on normalized data.

| | array | sampleNames | *1 | *2 | *3 | Group | Genotype | Temperature | ShortName |
|---|---|---|---|---|---|---|---|---|---|
| ☐ | 1 | WT.RT.1 | | | | WT.RT | WT | RT | WT.RT.1 |
| ☐ | 2 | WT.RT.2 | | | | WT.RT | WT | RT | WT.RT.2 |
| ☐ | 3 | WT.RT.3 | | | | WT.RT | WT | RT | WT.RT.3 |
| ☐ | 4 | KO.RT.1 | | | | KO.RT | KO | RT | KO.RT.1 |
| ☐ | 5 | KO.RT.2 | | | | KO.RT | KO | RT | KO.RT.2 |
| ☐ | 6 | KO.RT.3 | | | | KO.RT | KO | RT | KO.RT.3 |
| ☐ | 7 | WT.COLD.1 | | | | WT.COLD | WT | COLD | WT.COLD.1 |
| ☐ | 8 | WT.COLD.2 | | | | WT.COLD | WT | COLD | WT.COLD.2 |
| ☐ | 9 | WT.COLD.3 | | | | WT.COLD | WT | COLD | WT.COLD.3 |
| ☐ | 10 | KO.COLD.1 | x | x | | KO.COLD | KO | COLD | KO.COLD.1 |
| ☐ | 11 | KO.COLD.2 | | | | KO.COLD | KO | COLD | KO.COLD.2 |
| ☐ | 12 | KO.COLD.3 | | | | KO.COLD | KO | COLD | KO.COLD.3 |

**Fig. 6** Aspect of the summary table, in the index.html file, produced by the arrayQualityMetrics package on normalized data

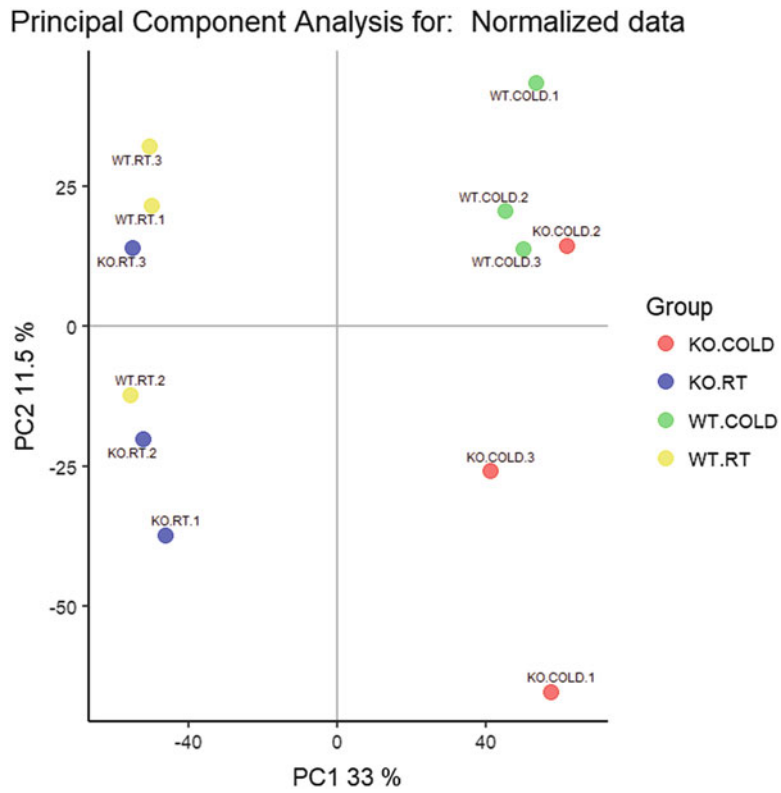


**Fig. 7** Visualization of first two principal components for normalized data

```
> plotPCA3(exprs(eset_rma), labels = targets$ShortName, factor
= targets$Group,
+          title="Normalized data", scale = FALSE, size = 3,
+          colores = c("red", "blue", "green", "yellow"))
```

Now first component accounts for 33% of the total variability. Notice that the percentage of explained variability has decreased with respect to PCA performed on raw data. As similar as the PCA with raw data, it separates samples from *COLD* level of *temperature* condition on the right, and samples from *RT* level on the left. It is important to note that there are one sample from group *KO.RT* that groups near **WT.RT** and vice versa. It could be an issue of mislabeling of samples that should be checked with the laboratory that has processed the samples.

Figure 8 shows a multiple boxplot depicting the distribution of the normalized intensities along all samples. Notice that all boxplots have the same aspect. This suggests that the normalization has worked fine. However, it is important to be aware that RMA includes a step ("quantile normalization") where the empirical distribution of all the samples is set to the same values. As a

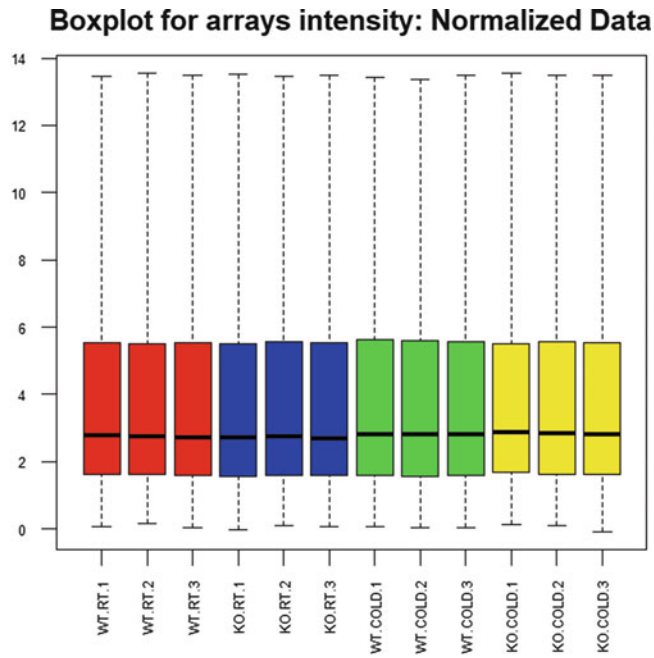


**Fig. 8** Distribution of intensities for normalized data

consequence, it is expected that the boxplots are identical or at least very similar.

```
> boxplot(eset_rma, cex.axis=0.5, las=2, which="all",
+         col = c(rep("red", 3), rep("blue", 3), rep("green", 3),
rep("yellow", 3)),
+          main="Boxplot for arrays intensity: Normalized Data")
```

### 3.8 Batch Detection

Gene expression microarray results can be affected by minuscule differences in any number of nonbiological variables like reagents from different lots, different technicians, and, the more usual issue, different processing dates of samples from the same experiment. The cumulative error introduced by these time and place-dependent experimental variations is referred to as "batch effects." Different approaches have been developed for identifying and removing batch effects from microarray data like surrogate variable analysis, Combat, and principal variation component analysis (PVCA).

Here we will use the last one, principal variation component analysis, which estimates source and proportion of variation in two steps, principal component analysis and variance component analysis. Only for illustration purposes we have added a new column to our targets file, with a fictitious sample processing date. We will perform the PVCA analysis before and after adding this column to see the differences.

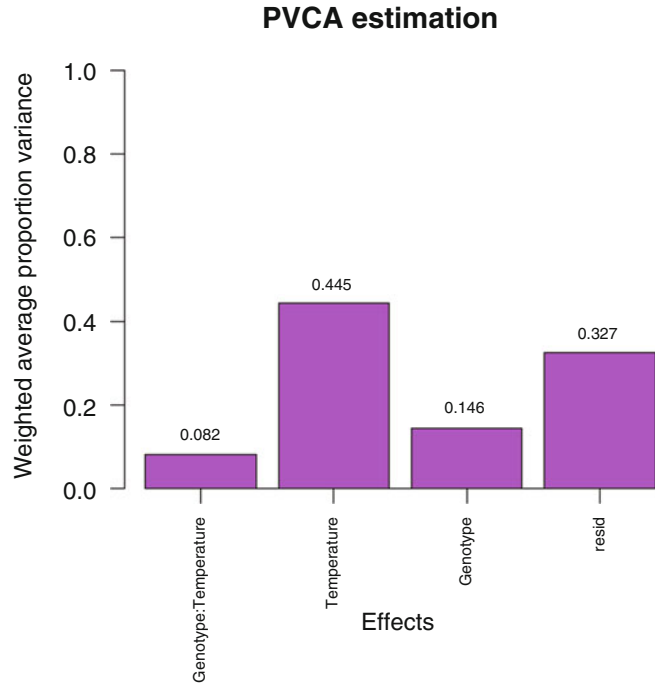


**Fig. 9** Relative importance of the different factors—genotype, temperature, and interaction—affecting gene expression

```
> #load the library
> require(pvca)
> pData(eset_rma) <- targets
> #select the threshold
> pct_threshold <- 0.6
> #select the factors to analyze
> batch.factors <- c("Genotype", "Temperature")
> #run the analysis
> pvcaObj <- pvcaBatchAssess (eset_rma, batch.factors,
pct_threshold)
```

Figure 9 shows a bar diagram with one bar per each source of variation included in the analysis. Their relative size indicates the percentage of variability attributable to each source. The plot shows that the main source of variation in the samples is the *Temperature* condition. This was also observed on the PCA plots on raw and normalized data in Figs. 4 and 7.

```
> #plot the results
> bp <- barplot(pvcaObj$dat, xlab = "Effects",
+   ylab = "Weighted average proportion variance",
+   ylim= c(0,1.1),col = c("mediumorchid"), las=2,
+   main="PVCA estimation")
```

```
> axis(1, at = bp, labels = pvcaObj$label, cex.axis = 0.55,
las=2)
> values = pvcaObj$dat
> new_values = round(values , 3)
> text(bp,pvcaObj$dat,labels = new_values, pos=3, cex = 0.5)
```

### 3.9 Detecting Most Variable Genes

Selection of differentially expressed genes is affected by the number of genes on which we make it. The higher the number, the greater the necessary adjustment of *p*-values (as will be seen below), which will lead to end up miscarrying more genes.

If a gene is differentially expressed, it is expected that there is a certain difference between the groups, and therefore the overall variance of the gene will be greater than that of those that do not have differential expression. Plotting the overall variability of all genes is useful to decide which percentage of genes shows a variability that can be attributed to other causes than random variation. Figure 10 depicts the standard deviations of all genes sorted from smallest to biggest values. The plot shows that the most variable genes are those with a standard deviation above 90–95% of all standard deviations.

```
> sds <- apply (exprs(eset_rma), 1, sd)
> sdsO<- sort(sds)
> plot(1:length(sdsO), sdsO, main="Distribution of variability
for all genes",
```

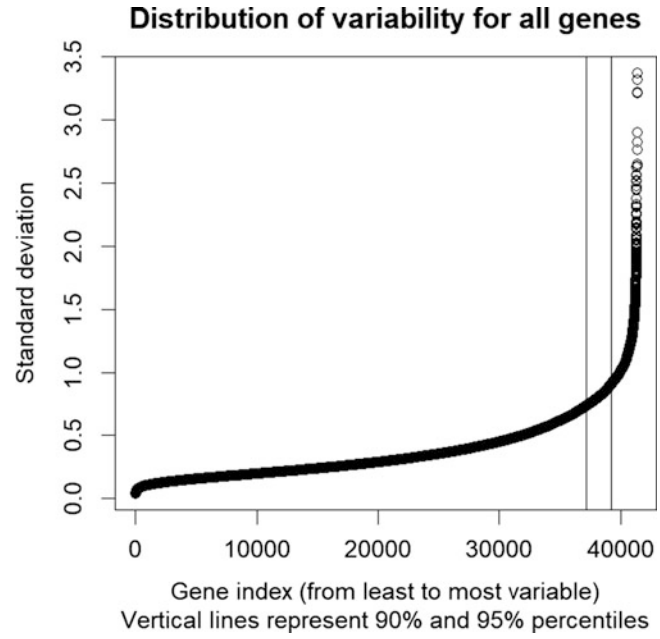


**Fig. 10** Values of standard deviations along all samples for all genes ordered from smallest to biggest

```
+      sub="Vertical lines represent 90% and 95% percentiles",
+         xlab="Gene index (from least to most variable)",
ylab="Standard deviation")
> abline(v=length(sds)*c(0.9,0.95))
```

### 3.10 Filtering Least Variable Genes

Filtering out those genes whose variability can be attributed to random variation, that is, the genes that are, reasonably, not expected to be differentially expressed, has proven to be useful to reduce the number of tests to be performed with the corresponding increase in power [23].

Function nsFilter from the bioconductor package genefilter can be used to remove genes based on a variability threshold. If an annotation package—associating probesets identifiers and gene identifiers from different databases—is available it can also be used to remove probesets which do not have a gene identifier associated.

```
> require(genefilter)
> require(mogene21sttranscriptcluster.db)
> annotation(eset_rma) <- "mogene21sttranscriptcluster.db"
> filtered <- nsFilter(eset_rma,
+                      require.entrez = TRUE, remove.dupEntrez = TRUE,
+                      var.filter=TRUE, var.func=IQR, var.cutoff=0.75,
+                     filterByQuantile=TRUE, feature.exclude = "^AFFX")
```

Function nsFilter returns the filtered values and a report of the filtering results.

```
> print(filtered$filter.log)
$numDupsRemoved
[1] 672

$numLowVar
[1] 17994

$numRemoved.ENTREZID
[1] 16681
> eset_filtered <-filtered$eset
```

After filtering there are 5998 genes left. Note that we have stored the genes left in the variable *eset_filtered*.

### 3.11 Saving Normalized and Filtered Data

Normalized filtered data are the starting point for further analyses but we may want to go back to them, for example to review specific gene expression values. It is usual to save the binary objects but also to write expression values into text or excel files. Writing to Excel from R is not a trivial task—for strange it may seem—because different packages work differently depending of the operating system, so it is omitted from the code.

```
> write.csv(exprs(eset_rma), file="./results/normalized.Data.csv")
> write.csv(exprs(eset_filtered),
file="./results/normalized.Filtered.Data.csv")
> save(eset_rma, eset_filtered, file="./results/normalized.Data.Rda")
```

### 3.12 Defining the Experimental Setup: The Design Matrix

Selection of differentially expressed genes basically consists of doing some type of test, usually on a gene-wise basis, to compare gene expression between groups. This can be done using many different approaches (*see* [24]). There is a general agreement that using standard statistical tests such as *t*-tests is not appropriate [25] and that better options are methods that perform some type of variance shrinking [26]. Techniques specifically developed for microarrays such as SAM [27] or Linear Models for Microarrays [28] have proved to produce much better results [24].

In this protocol the Linear Models for Microarrays method, implemented in the limma package [14] is used to select differentially expressed genes.

The first step for the analysis based on linear models is to create the **design matrix**. Basically, it is a table that describes the allocation of each sample to a group or experimental condition. It has as many rows as samples and as many columns as groups (if only one factor is considered). Each row contains a one in the column of the group to which the sample belongs and a zero in the others.

The design matrix can be defined manually or from a factor variable that may have been introduced in the "targets" file with this aim created specifically for it. In this study that "Group" variable is a combination of the two experimental conditions, "KO/Wild" and "RT/COLD" which are jointly represented as one factor with four levels.

```
> if (!exists("eset_filtered")) load
(file="./results/normalized.Data.Rda")
> require(limma)
> designMat<- model.matrix(~0+Group, pData(eset_filtered))
> colnames(designMat) <- c("KO.COLD", "KO.RT", "WT.COLD", "WT.RT")
> print(designMat)
   KO.COLD KO.RT WT.COLD WT.RT
1        0     0       0     1
2        0     0       0     1
3        0     0       0     1
4        0     1       0     0
5        0     1       0     0
6        0     1       0     0
7        0     0       1     0
8        0     0       1     0
9        0     0       1     0
10       1     0       0     0
```

```
11        1     0        0      0
12        1     0        0      0
attr(,"assign")
[1] 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$Group
[1] "contr.treatment"
```

### 3.13 Defining Comparisons with the Contrast Matrix

The contrast matrix is used to describe the comparisons between groups. It consists of as many columns as comparisons and as many rows as groups (i.e., as columns of the design matrix). A comparison between groups—called "contrast"—is represented by a "1" and a "-1" in the rows of groups to compare and zeros in the rest. If several groups are involved in the comparison we would have as many coefficients as groups with the only restriction that its sum would be zero.

In this example we want to check the effect of knocking out a gene ("KO vs WT") separately for cold and RT temperature. Also, we want to test if there is any interaction between knocking out the gene and temperature. This can be done by doing three comparisons described below:

```
> cont.matrix <- makeContrasts (KOvsWT.COLD = KO.COLD-WT.COLD,
+                               KOvsWT.RT = KO.RT-WT.RT,
+                               INT = (KO.COLD-WT.COLD) - (KO.RT-WT.RT),
+                               levels=designMat)
> print(cont.matrix)
         Contrasts
Levels    KOvsWT.COLD KOvsWT.RT INT
  KO.COLD           1         0   1
  KO.RT             0         1  -1
  WT.COLD          -1         0  -1
  WT.RT             0        -1   1
```

The contrast matrix is defined to perform three comparisons: effect of KO in Cold temperature, effect of KO in RT temperature, and interaction between KO and temperature.

### 3.14 Model Estimation and Gene Selection

Once the design matrix and the contrasts have been defined, we can proceed to estimate the model, estimate the contrasts and perform the significance tests that will lead to the decision, for each gene and each comparison, if they can be considered differentially expressed.

The method implemented in the package extends the traditional analysis using Empirical Bayes models to combine an estimate of variability based on the entire matrix with individual estimates based on each individual value providing improved error estimates [28].

The analysis provides the usual test statistics such as Fold-change, *t*-moderated or adjusted *p*-values that are used to order the genes from more to less differential expressed.

In order to control the percentage of false positives that may result from high number of contrasts made simultaneously the *p*-values are adjusted so that we have control over the false positive rate using the Benjamini and Hochberg method [29].

All relevant information for further exploration of the results is stored in an R object of class MArrayLM defined in the limma package. Here it is named as fit.main.

```
> require(limma)
> fit<-lmFit(eset_filtered, designMat)
> fit.main<-contrasts.fit(fit, cont.matrix)
> fit.main<-eBayes(fit.main)
> class(fit.main)
[1] "MArrayLM"
attr(,"package")
[1] "limma"
```

### 3.15 Obtaining Lists of Differentially Expressed Genes

The package implements function topTable which contains, for a given contrast a list of genes ordered from smallest to biggest *p*-value which can be considered from most to least differentially expressed. For each gene the following statistics are provided:

We can have a look at the first lines of each topTable.

For comparison 1 (KOvsWT.COLD): Genes that change their expression between KO and WT in cold temperature:

```
> topTab_KOvsWT.COLD <- topTable (fit.main, number=nrow(fit.
main), coef="KOvsWT.COLD", adjust="fdr")
> head(topTab_KOvsWT.COLD)
```

| | logFC | AveExpr | t | P.Value | adj.P.Val | B |
|---|---|---|---|---|---|---|
| 17497829 | 2.474954 | 6.706683 | 15.73346 | 0e+00 | 0.0000302 | 10.274342 |
| 17407049 | 3.676524 | 4.697071 | 14.65945 | 0e+00 | 0.0000324 | 9.720840 |
| 17529307 | −3.628616 | 3.493648 | −10.96795 | 2e-07 | 0.0003533 | 7.266518 |
| 17373930 | 2.071352 | 5.412865 | 10.79138 | 3e-07 | 0.0003533 | 7.122736 |
| 17251565 | 1.991109 | 2.955199 | 10.72126 | 3e-07 | 0.0003533 | 7.064847 |
| 17291821 | −2.434786 | 5.348719 | −10.24479 | 5e-07 | 0.0004045 | 6.659254 |

For comparison 2 (KOvsWT.RT): Genes that change their expression between KO and WT in room temperature:

```
> topTab_KOvsWT.RT <- topTable (fit.main, number=nrow(fit.
main), coef="KOvsWT.RT", adjust="fdr")
> head(topTab_KOvsWT.RT)
```

| | logFC | AveExpr | t | P.Value | adj.P.Val | B |
|---|---|---|---|---|---|---|
| 17407049 | 3.675590 | 4.697071 | 14.65573 | 0e+00 | 0.0000243 | 9.811319 |
| 17282970 | −3.565316 | 9.093793 | −14.60418 | 0e+00 | 0.0000243 | 9.782552 |
| 17497829 | 2.280728 | 6.706683 | 14.49875 | 0e+00 | 0.0000243 | 9.723258 |
| 17425609 | 2.207391 | 2.464984 | 11.91994 | 1e-07 | 0.0001205 | 8.053747 |
| 17472497 | −3.366076 | 7.132534 | −11.88661 | 1e-07 | 0.0001205 | 8.029086 |
| 17399411 | −1.816331 | 4.457054 | −11.15112 | 2e-07 | 0.0001891 | 7.461733 |

For comparison 3 (INT): Genes that behave differently between comparison 1 and 2:

```
> topTab_INT <- topTable (fit.main, number=nrow(fit.main),
coef="INT", adjust="fdr")
> head(topTab_INT)
```

| | logFC | AveExpr | t | P.Value | adj.P.Val | B |
|---|---|---|---|---|---|---|
| 17282970 | 3.369838 | 9.093793 | 9.760524 | 8.00e-07 | 0.0046281 | 5.715626 |
| 17494820 | 2.023908 | 1.704030 | 8.447940 | 3.30e-06 | 0.0098332 | 4.580929 |
| 17251565 | 2.127194 | 2.955199 | 8.099213 | 5.00e-06 | 0.0099083 | 4.243694 |
| 17274184 | −1.800778 | 8.001305 | −7.330313 | 1.29e-05 | 0.0180537 | 3.440917 |
| 17543045 | 2.012296 | 9.748289 | 7.212880 | 1.50e-05 | 0.0180537 | 3.310756 |
| 17432247 | 2.798819 | 4.218885 | 7.054483 | 1.85e-05 | 0.0185289 | 3.131888 |

First column of each topTable contains the manufacturer's (Affymetrix) ID for each probeset. Next step is to guess which gene correspond to each Affymetrix ID. This process is called **annotation**.

### 3.16 Gene Annotation

Once we have the top table it is useful to provide additional information on the features that have been selected. This process is called "annotation" and essentially what it does is to look for information to associate identifiers that appear in the top table, usually corresponding to probesets or transcripts depending of the array type, with more familiar names such as the Gene Symbol, the Entrez Gene identifier, or the Gene description.

For simplicity, because there are three topTables, a function annotating one topTable with a given package is prepared and used.

```
> annotatedTopTable <- function(topTab, anotPackage)
+ {
+   topTab <- cbind(PROBEID=rownames(topTab), topTab)
+   myProbes <- rownames(topTab)
+   thePackage <- eval(parse(text = anotPackage))
+  geneAnots <- select(thePackage, myProbes, c("SYMBOL", "ENTREZID",
"GENENAME"))
+    annotatedTopTab<- merge(x=geneAnots, y=topTab, by.x="PROBEID",
by.y="PROBEID")
+ return(annotatedTopTab)
+ }
> topAnnotated_KOvsWT.COLD <- annotatedTopTable(topTab_KOvsWT.COLD,
+ anotPackage="mogene21sttranscriptcluster.db")
> topAnnotated_KOvsWT.RT <- annotatedTopTable(topTab_KOvsWT.RT,
+ anotPackage="mogene21sttranscriptcluster.db")
> topAnnotated_INT <- annotatedTopTable(topTab_INT,
+ anotPackage="mogene21sttranscriptcluster.db")
> write.csv(topAnnotated_KOvsWT.COLD, file="./results/topAnnotated_-
KOvsWT_COLD.csv")
> write.csv(topAnnotated_KOvsWT.RT, file="./results/topAnnotated_-
KOvsWT_RT.csv")
> write.csv(topAnnotated_INT, file="./results/topAnnotated_INT.csv")
```

Annotation makes the tables more comprehensible. Table 3 shows the annotations added to results "topTable" for the comparison "KOvsWT.COLD" (only the first four columns are shown).

```
> short <- head(topAnnotated_KOvsWT.COLD[1:5,1:4])
> knitr::kable(
+   short, booktabs = TRUE,
+   caption = 'Annotations added to results "topTable" for the
comparison "KOvsWT.COLD"'
+ )
```

**Table 3**
**Annotations added to results "topTable" for the comparison "KOvsWT.COLD"**

| PROBEID | SYMBOL | ENTREZID | GENENAME |
|---|---|---|---|
| 17210887 | Atp6v1h | 108664 | ATPase, H+ transporting, lysosomal V1 subunit H |
| 17210984 | Pcmtd1 | 319263 | protein-L-isoaspartate (D-aspartate) O-methyltransferase domain containing 1 |
| 17211000 | Rrs1 | 59014 | ribosome biogenesis regulator 1 |
| 17211131 | Prex2 | 109294 | phosphatidylinositol-3,4,5-trisphosphate-dependent Rac exchange factor 2 |
| 17211174 | A830018L16Rik | 320492 | RIKEN cDNA A830018L16 gene |

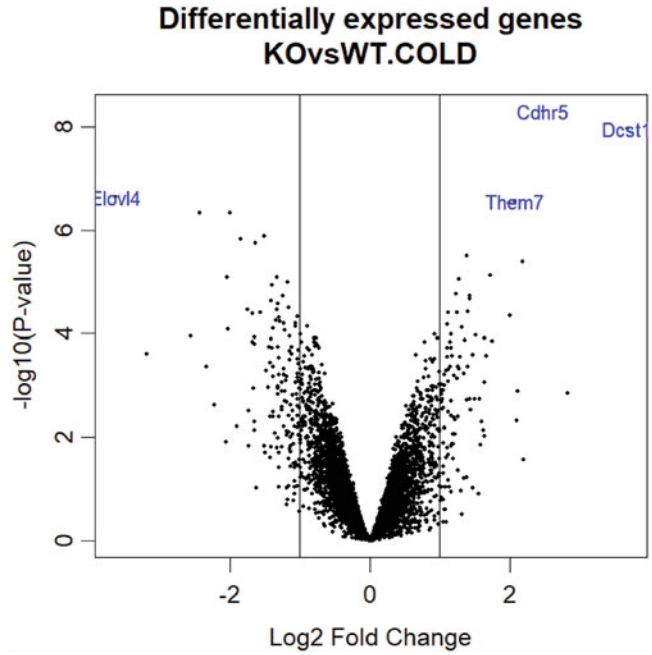


**Fig. 11** Volcano plot for the comparison between KO and WT in COLD temperature. The names of the top 4 genes (i.e., the first four genes in the topTable) are shown in the plot

### 3.17 Visualizing Differential Expression

A visualization of the overall differential expression can be obtained using volcano-plots. These plots show if there are many or few genes with a large fold-change and significantly expressed or if this number is low. These graphs represent in the X-axis the changes of expression in logarithmic scale ("biological effect") and in the Y-axis the "minus logarithm" of the *p*-value or alternatively the B statistic ("Statistical effect"). Figure 11 shows a volcano plot for the comparison between KO and WT in COLD temperature. The names of the top ten genes (i.e., the first ten genes in the topTable) are shown in the plot.

```
> require(mogene21sttranscriptcluster.db)
> geneSymbols <- select(mogene21sttranscriptcluster.db, row-
names(fit.main), c("SYMBOL"))
> SYMBOLS<- geneSymbols$SYMBOL
> volcanoplot(fit.main, coef=1, highlight=10, names=SYMBOLS,
+                main=paste("Differentially expressed genes",
colnames(cont.matrix)[1], sep="\n"))
> abline(v=c(-1,1))
```

### 3.18 Multiple Comparisons

When one selects genes in several comparisons it is usually interesting to know which genes have been selected in each comparison. Sometimes, biologically relevant genes will be those that are

selected in one of them but not in others. On other occasions, the interest will lie with genes that are selected in all comparisons.

Functions decideTests and VennDiagram from package limma can be used to annotate and count the genes selected in every comparison.

```
> require(limma)
> res<-decideTests(fit.main, method="separate", adjust.meth-
od="fdr", p.value=0.1, lfc=1)
```

This object has as many columns as comparisons and as many rows as genes. Per each gene and comparison a "+1" denotes significantly upregulated ($t$-test values $>0$, FDR $<$ selected cutoff), a "-1" significantly downregulated ($t$-test values $<0$, FDR $<$ selected cutoff), and a "0" nonsignificant difference (FDR $>$ selected cutoff).

```
> sum.res.rows<-apply(abs(res),1,sum)
> res.selected<-res[sum.res.rows!=0,]
> print(summary(res))
   KOvsWT.COLD KOvsWT.RT  INT
-1          99        72   24
0         5830      5846 5953
1           69        80   21
```

This can be visualized in a Venn Diagram. Figure 12 shows a Venn diagram depicting the number of genes that have been called differentially expressed in each comparison with a given cutoff (here the cutoff is defined by "FDR $<$ 0.1" and "logFC $>$ 1" The figure shows how many of these genes are shared by one or more selections.

```
> vennDiagram (res.selected[,1:3], cex=0.9)
> title("Genes in common between the three comparisons\n Genes
selected with FDR < 0.1 and logFC > 1")
```

### 3.19 Heatmaps

Genes that have been selected as differentially expressed may be visualized using a heatmap. These plots use color palettes to highlight distinct values—here positive (upregulation) or negative (downregulation) significantly different expressions.

*Heatmaps* can be used to visualize the expression values of differentially expressed genes with no specific order, but it is usually preferred to plot them doing a hierarchical clustering on genes (rows) or columns(samples) in order to find groups of genes with common patterns of variation which can eventually be associated to the different groups being compared.

There may be discussion on which genes to select for doing a heatmap. A common option is to select the gens that have been

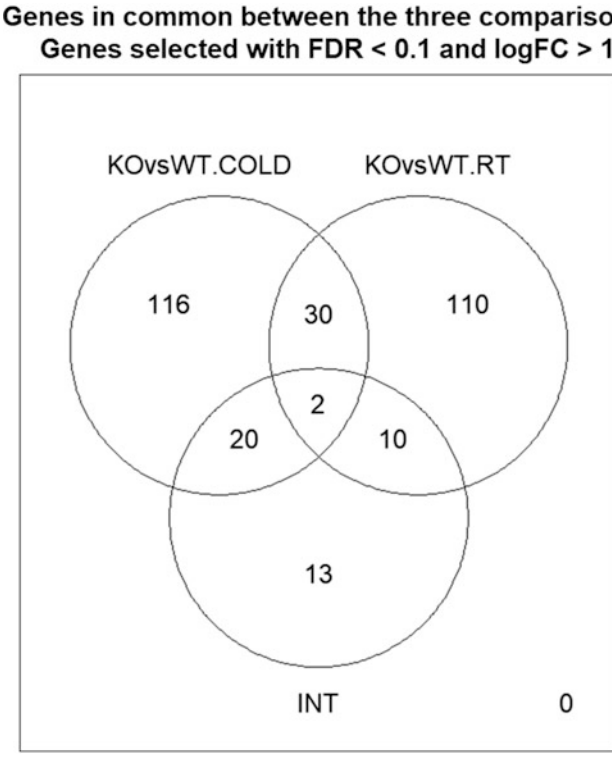


**Fig. 12** Venn diagram showing the genes in common between the three comparisons performed

selected in the previous steps, that is, the genes that have been called differentially expressed in at least one comparison.

```
> probesInHeatmap <- rownames(res.selected)
> HMdata <- exprs(eset_filtered)[rownames(exprs(eset_
filtered)) %in% probesInHeatmap,]
>
> geneSymbols <- select(mogene21sttranscriptcluster.db,
rownames(HMdata), c("SYMBOL"))
> SYMBOLS<- geneSymbols$SYMBOL
> rownames(HMdata) <- SYMBOLS
> write.csv(HMdata, file = file.path("./results/data4Heatmap.
csv"))
```

With the selected data a heatmap can be generated with or without clustering genes and/or samples.

Figure 13 shows a heatmap produced for all the genes selected with the same criteria described above (FDR $< 0.1$ and logFC $> 1$) where no clustering of genes and samples is performed.

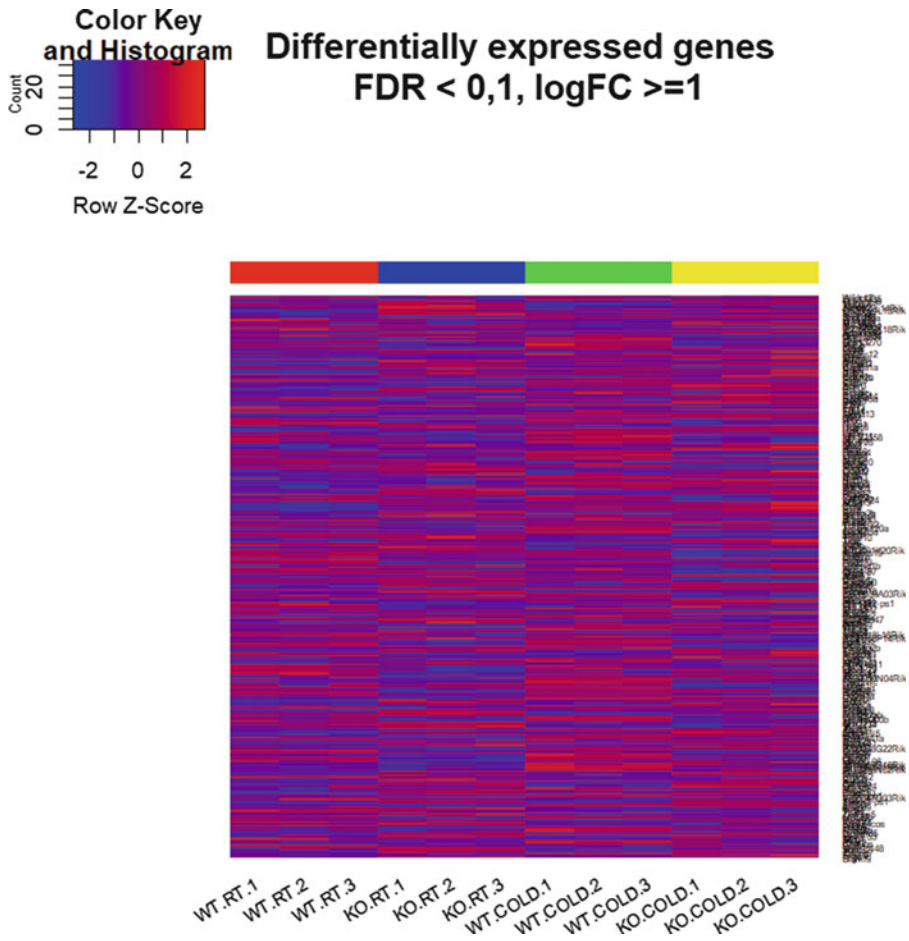


**Fig. 13** Heatmap for expression data without any grouping

```
> my_palette <- colorRampPalette(c("blue", "red"))(n = 299)
> require(gplots)
>
> heatmap.2(HMdata,
+           Rowv = FALSE,
+           Colv = FALSE,
+         main = "Differentially expressed genes \n FDR < 0,1,
logFC >=1",
+           scale = "row",
+           col = my_palette,
+           sepcolor = "white",
+           sepwidth = c(0.05,0.05),
+           cexRow = 0.5,
+           cexCol = 0.9,
+           key = TRUE,
+           keysize = 1.5,
+           density.info = "histogram",
```

```
+           ColSideColors = c(rep("red",3),rep("blue",3), rep
("green",3), rep("yellow",3)),
+           tracecol = NULL,
+           dendrogram = "none",
+           srtCol = 30)
```

Figure 14 shows a heatmap produced for all the genes selected with the same criteria described above (FDR < 0.1 and logFC > 1) where genes and samples are forced to group by row and column similarity, respectively.

```
> heatmap.2(HMdata,
+           Rowv = TRUE,
+           Colv = TRUE,
+           dendrogram = "both",
+         main = "Differentially expressed genes \n FDR < 0,1,
logFC >=1",
```

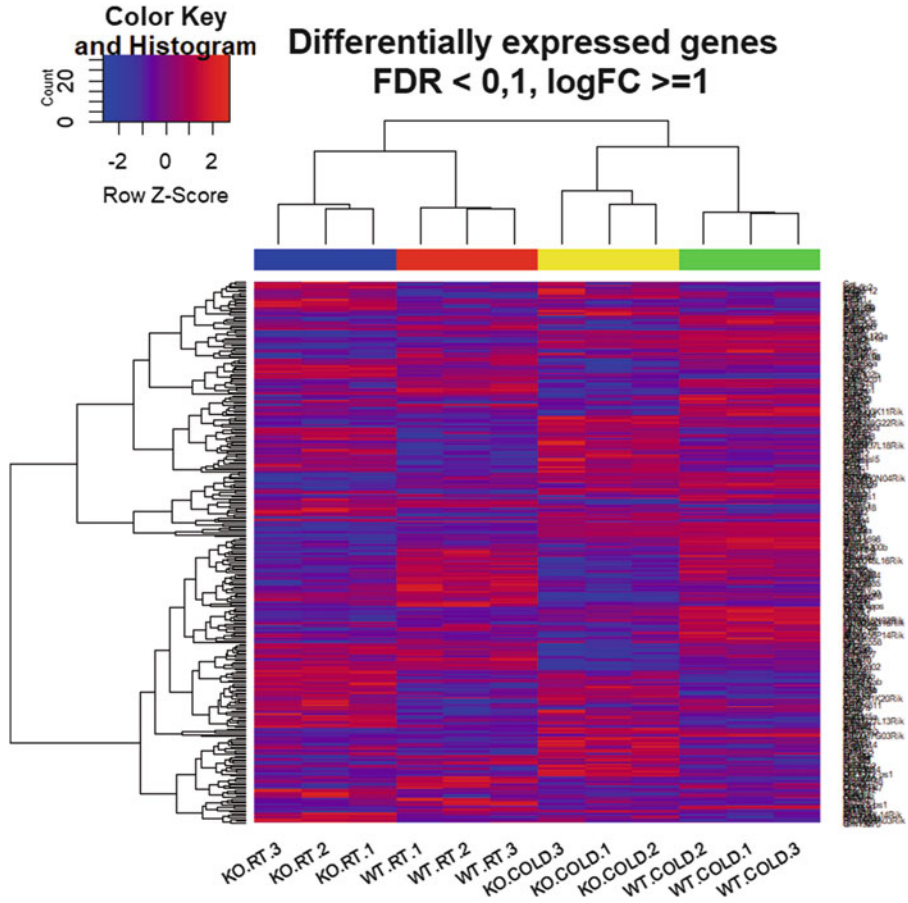


**Fig. 14** Heatmap for expression data grouping genes (rows) and samples (columns) by their similarity

```
+           scale = "row",
+           col = my_palette,
+           sepcolor = "white",
+           sepwidth = c(0.05,0.05),
+           cexRow = 0.5,
+           cexCol = 0.9,
+           key = TRUE,
+           keysize = 1.5,
+           density.info = "histogram",
+           ColSideColors = c(rep("red",3),rep("blue",3), rep
("green",3), rep("yellow",3)),
+           tracecol = NULL,
+           srtCol = 30)
```

### 3.20 Biological Significance of Results

Once a list of gene has been obtained that characterizes the difference between two conditions it has to be interpreted. Although this requires, of course, a good understanding of the underlying biological problem, a statistical approach known as "Gene Set Analysis" can be useful for suggesting ideas for the interpretation.

With this aim these types of analyses seek to establish whether, given a list of genes selected for being differentially expressed between two conditions, the functions, biological processes, or molecular pathways that characterize them appear on this list more frequently than among the rest of the genes analyzed.

There are many variants of these types of analysis (*see* [30]), but here we will use the basic enrichment analysis as described and implemented in the ReactomePA Bioconductor package. The analysis is done on the ReactomePA annotation database https://reactome.org/.

Analyses of this type need a minimum number of genes to be reliable, preferably a few hundreds than a few dozens, so it is common to perform a selection less restrictive than with the previous steps. For instance, an option is to include all genes with a nonstringent FDR cutoff, such as FDR < 0.15 without filtering by minimum "fold-change").

In the first step we prepare the list of gene lists that will be analyzed:

```
> listOfTables <- list(KOvsWT.COLD = topTab_KOvsWT.COLD,
+                      KOvsWT.RT  = topTab_KOvsWT.RT,
+                      INT = topTab_INT)
> listOfSelected <- list()
> for (i in 1:length(listOfTables)){
+   # select the toptable
+   topTab <- listOfTables[[i]]
+   # select the genes to be included in the analysis
+   whichGenes<-topTab["adj.P.Val"]<0.15
+   selectedIDs <- rownames(topTab)[whichGenes]
```

```
+   # convert the ID to Entrez
+   EntrezIDs<- select(mogene21sttranscriptcluster.db, selec-
tedIDs, c("ENTREZID"))
+   EntrezIDs <- EntrezIDs$ENTREZID
+   listOfSelected[[i]] <- EntrezIDs
+   names(listOfSelected)[i] <- names(listOfTables)[i]
+ }
> sapply(listOfSelected, length)
KOvsWT.COLD   KOvsWT.RT         INT
        759         452          85
```

The analysis also requires to have the Entrez Identifiers for all the genes analyzed.

```
> EntrezUni <- topAnnotated_KOvsWT.COLD$ENTREZID
```

The biological significance analysis will be applied only to the first two lists. Sometimes, yet another decomposition is applied so that upregulated and downregulated genes are separately analyzed. This will not be done here because there is no clear biological argument to proceed so in all cases.

```
> require(ReactomePA)
>
> listOfData <- listOfSelected[1:2]
> comparisonsNames <- names(listOfData)
>
> organisme <- "mouse"
> universe <- as.character(EntrezUni)
>
> for (i in 1:length(listOfData)){
+   data <- listOfData[[i]]
+   genesIn <- listOfSelected[[i]]
+   comparison <- comparisonsNames[i]
+   enrich.result <- enrichPathway(gene = genesIn,
+                                  pvalueCutoff = 0.05,
+                                  readable = T,
+                                  organism =  organisme,
+                                  universe = universe,
+                                  minGSSize = 5,
+                                  maxGSSize = 500,
+                                  pAdjustMethod = "BH")
+
+   if (length(rownames(enrich.result@result)) != 0) {
+   write.csv(as.data.frame(enrich.result),
+             file =paste0("./results/","ReactomePA.Results.",
comparison,".csv"),
+              row.names = FALSE)
```

```
+
+    pdf(file=paste0("./results/","ReactomePABarplot.",compar-
ison,".pdf"))
+     print(barplot(enrich.result, showCategory = 15, font.size
= 4,
+             title = paste0("Reactome Pathway Analysis for ",
comparison,". Barplot")))
+    dev.off()
+
+    pdf(file = paste0("./results/","ReactomePAcnetplot.",com-
parison,".pdf"))
+      print(cnetplot(enrich.result, categorySize = "geneNum",
schowCategory = 15,
+          vertex.label.cex = 0.75))
+    dev.off()
+    }
+ }
IGRAPH 8f7cba7 UN-- 9 8 --
+ attr: name (v/c), size (v/n), color (v/c), label (v/c), width
| (e/n), color (e/c)
+ edges from 8f7cba7 (vertex names):
[1] Synthesis of very long-chain fatty acyl-CoAs--Elovl4
[2] Synthesis of very long-chain fatty acyl-CoAs--Elovl2
[3] Synthesis of very long-chain fatty acyl-CoAs--Acsl5
[4] Synthesis of very long-chain fatty acyl-CoAs--Elovl7
[5] Synthesis of very long-chain fatty acyl-CoAs--Elovl3
[6] Synthesis of very long-chain fatty acyl-CoAs--Acsl4
[7] Synthesis of very long-chain fatty acyl-CoAs--Hsd17b12
+ ... omitted several edges
```

The results obtained in the analysis of biological significance are the following:

- A *.csv* file with a summary of all the enriched pathways and the associated statistics.
- A bar plot with the best enriched pathways. Height of the bar plot is the number of genes of our analysis related with that pathway. Moreover, pathways are ordered by statistical significance.
- A plot with a network of the enriched pathways and the relation among the genes included.

In our study, for comparison *KOvsWT.COLD* only a single enriched pathway has been found, *Synthesis of very long-chain fatty acyl-CoAs*, while for comparison *KOvsCTL.RT*, four enriched

**Table 4**
**First rows and columns for Reactome results on KOvsWT.RT.csv comparison**

| | Description | GeneRatio | BgRatio | pvalue | p.adjust |
|---|---|---|---|---|---|
| R-MMU-1483249 | Inositol phosphate metabolism | 7/200 | 16/2422 | 0.0001418 | 0.038842 |
| R-MMU-166658 | Complement cascade | 6/200 | 13/2422 | 0.0003086 | 0.038842 |
| R-MMU-977606 | Regulation of Complement cascade | 6/200 | 13/2422 | 0.0003086 | 0.038842 |
| R-MMU-1855204 | Synthesis of IP3 and IP4 in the cytosol | 5/200 | 9/2422 | 0.0003499 | 0.038842 |

pathways have been found (Table 4). The more statistically relevant is _ Inositol phosphate metabolism_.

```
> Tab.react <- read.csv2(file.path("./results/ReactomePA.Results.KOvsWT.RT.csv"),
+                        sep = ",", header = TRUE, row.names = 1)
>
> Tab.react <- Tab.react[1:4, 1:5]
> knitr::kable(Tab.react, booktabs = TRUE, caption = "First rows and columns for Reactome results on KOvsWT.RT.csv comparison")
```

### 3.21 Summary of Results

Once the process has been completed, one has to obtained a, sometimes long, list of files with the data and the analysis results. These files are of the basis for discussing the results and looking for a biological interpretation. Both aspects are beyond the goals of this chapter, so they are omitted here.

It is useful to create a file with the type, name and description of all the files generated along the analysis. Table 5 shows the list of files generated in the current case study.

```
> listOfFiles <- read.table(file="results/listOfFiles.txt", sep="\t", head=T)
> knitr::kable(
+   listOfFiles, booktabs = TRUE,
+   caption = 'List of files generated in the analysis'
+ )
```

**Table 5**
**List of files generated in the analysis**

| File | Category | Order | Description |
|---|---|---|---|
| targets.csv | DATA | 1 | Sample groups, labels and covariates information file |
| normalized.Data.csv | DATA | 2 | Normalized expression values for all genes (or transcripts, or probesets) |
| normalized.Filtered.Data.csv | DATA | 3 | Normalized expression values for genes (or transcripts, or probesets) |
| BoxplotRaw.pdf | QC | 4 | Intensity distribution for raw data |
| BoxplotNorm.pdf | QC | 5 | Intensity distribution for normalized data |
| PCAraw.pdf | QC | 6 | Principal Component Analysis plot for raw data |
| PCANorm.pdf | QC | 7 | Principal Component Analysis plot for normalized data |
| QCDir.Raw/index.html | QC | 8 | Quality control plots of raw data with ArrayQualityMetrics package |
| QCDir.Norm/index.html | QC | 9 | Quality control plots of normalized data with ArrayQualityMetrics package |
| SDplot.pdf | ANALYSIS | 9 | Standard deviation of all genes (or transcripts or probesets) in the array |
| topAnnotated_KOvsCTL_COLD.csv | ANALYSIS | 10 | Top Table (with annotations) for the comparison: KOvsCTL in COLD temperature |
| topAnnotated_KovsCTL_RT.csv | ANALYSIS | 11 | Top Table (with annotations) for the comparison: KOvsCTL in ROOM temperature |
| topAnnotated_INT.csv | ANALYSIS | 12 | Top Table (with annotations) for the comparison between comparions: INT |
| Volcanos.pdf | ANALYSIS | 13 | Volcano plot for the comparisons performed |
| HeatmapUnordered.pdf | ANALYSIS | 14 | Heatmap made from genes selected from multiple comparisons. No additional ordering of rows. |
| HeatmapOrderByRows.pdf | ANALYSIS | 15 | Heatmap made from genes selected from multiple comparisons. Rows ordered by similarity |
| ReactomePA.Results.KovsCTL.COLD.csv | SIGBIO | 16 | Summary of all the enriched pathways and the associated statistics for the comparison: KOvsCTL in COLD temperature |
| ReactomePA.Results.KovsCTL.RT.csv | SIGBIO | 17 | Summary of all the enriched pathways and the associated statistics for the comparison: KOvsCTL in ROOM temperature |

(continued)

**Table 5**
**(continued)**

| File | Category | Order | Description |
|---|---|---|---|
| ReactomePA.Barplot.KovsCTL.COLD.pdf | SIGBIO | 18 | Bar plot with the best enriched pathways for the comparison: KOvsCTL in COLD temperature |
| ReactomePABarplot.KovsCTL.RT.pdf | SIGBIO | 19 | Bar plot with the best enriched pathways for the comparison: KOvsCTL in ROOM temperature |
| ReactomePA.cnetplot.KovsCTL.COLD.pdf | SIGBIO | 20 | Plot with a network of the enriched pathways and the relation among the genes included for the comparison: KOvsCTL in COLD temperature |
| ReactomePA.cnetplot.KovsCTL.RT.pdf | SIGBIO | 21 | plot with a network of the enriched pathways and the relation among the genes included for the comparison: KOvsCTL in ROOM temperature |

## 4 Notes

1. R and Bioconductor are open source software. This means that they are free (as in "free bier"), but it also means that compatibility between versions is not always 100% granted. It is important to know which version of R and Bioconductor is used for the analysis.
2. Although it may seem irrelevant it is important to be aware of regional settings before reading or writing text files. For instance, in some European countries the decimal point is the "comma," while in Anglo-Saxon ones it is the "dot." R can read any of these data formats, but, of course, it needs to be informed of which format is used in any situation.]
3. Sometimes, package installation may present some difficulties, often related to the operating system being used or even the workplace (e.g., if the user is behind a proxy). In these cases, the recommendation is to try to install packages one by one and, if needed contact the institution IT team.

### References

1. Efron B (2013) Large-scale inference: empirical Bayes methods for estimation, testing, and prediction. Cambridge University Press, Cambridge
2. Sánchez-Pla A (2014) DNA microarrays technology: overview and current status. In: Carolina Simó AC, García-Cañas V (eds) Comprehensive analytical chemistry, vol 63. Elsevier, pp 1–23
3. Draghici S (2012) Statistics and data analysis for microarrays using R and bioconductor. CRC Press, New York
4. Sánchez-Pla A, Reverter F, Ruíz de Villa MC, Comabella M (2012) Transcriptomics: mRNA

and alternative splicing. J Neuroimmunol 248:23–31. https://doi.org/10.1016/j.jneuroim.2012.04.008

5. Mehta JP, Rani S (2011) Software and tools for microarray data analysis. Methods Mol Biol 784:41–53
6. Carvalho BS, Irizarry RA (2010) A framework for oligonucleotide microarray preprocessing. Bioinformatics 26:2363–2367. https://doi.org/10.1093/bioinformatics/btq431
7. Carvalho B (2015) Pd.mogene.2.1.st: Platform design info for affymetrix mogene-2.1-st
8. Huber W, Carey VJ et al (2015) Orchestrating high-throughput genomic analysis with bioconductor. Nat Methods 12:115–121
9. Kauffmann A, Gentleman R, Huber W (2009) ArrayQualityMetrics–a bioconductor package for quality assessment of microarray data. Bioinformatics 25:415–416
10. Warnes GR, Bolker B, Bonebakker L, et al (2016) Gplots: various r programming tools for plotting data
11. Wickham H (2009) Ggplot2: elegant graphics for data analysis. Springer-Verlag, New York
12. Slowikowski K (2017) Ggrepel: repulsive text and label geoms for 'ggplot2'
13. Bushel P (2013) Pvca: principal variance component analysis (pvca)
14. Smyth GK (2005) limma: linear models for microarray data. In: Gentleman R, Carey V, Dudoit S, Irizarry R, Huber W (eds) Bioinformatics and computational biology solutions using r and bioconductor. Springer-Verlag, New York, pp 397–420
15. Gentleman R, Carey V, Huber W, Hahne F (2017) Genefilter: genefilter: methods for filtering genes from high-throughput experiments
16. Gentleman R (2017) Annotate: annotation for microarrays
17. Carlson M (2017) Org.Mm.eg.db: Genome wide annotation for mouse
18. MacDonald JW (2017) Mogene21sttranscriptcluster.db: Affymetrix mogene21 annotation data (chip mogene21sttranscriptcluster)
19. Yu G, He Q-Y (2016) ReactomePA: an R/Bioconductor package for reactome pathway analysis and visualization. Mol BioSyst 12:477–479. https://doi.org/10.1039/c5mb00663e
20. Li S, Mi L, Yu L et al (2017) Zbtb7b engages the long noncoding rna blnc1 to drive brown and beige fat development and thermogenesis. Proc Natl Acad Sci 114:E7111–E7120. https://doi.org/10.1073/pnas.1703494114
21. Clough E, Barrett T (2016) The Gene Expression Omnibus database. Methods Mol Biol 1418:93–110
22. Irizarry RA, Hobbs B, Collin F et al (2003) Exploration, normalization, and summaries of high density oligonucleotide array probe level data. Biostatistics 4:249–264. https://doi.org/10.1093/biostatistics/4.2.249
23. Hackstadt AJ, Hess AM (2009) Filtering for increased power for microarray data analysis. BMC Bioinformatics 10:11. https://doi.org/10.1186/1471-2105-10-11
24. Chrominski K, Tkacz M (2015) Comparison of high-level microarray analysis methods in the context of result consistency. PLoS One 10: e0128845. https://doi.org/10.1371/JOURNAL.PONE.0128845
25. Jeanmougin M, de Reynies A, Marisa L et al (2010) Should we abandon the t-Test in the analysis of gene expression microarray data: a comparison of variance modeling strategies. PLoS One 5:e12336. https://doi.org/10.1371/journal.pone.0012336
26. Allison DB, Cui X, Page GP, Sabripour M (2006) Microarray data analysis: from disarray to consolidation and consensus. Nat Rev Genet 7:55–65. https://doi.org/10.1038/nrg1749
27. Tusher VG, Tibshirani R, Chu G (2001) Significance analysis of microarrays applied to the ionizing radiation response. Proc Natl Acad Sci U S A 98:5116–5121. https://doi.org/10.1073/pnas.091062498
28. Smyth GK (2004) Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. Stat Appl Genet Mol Biol 3:1–25. https://doi.org/10.2202/1544-6115.1027
29. Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. J R Stat Soc Ser B Methodol 57:289–300
30. Khatri P, Sirota M, Butte AJ (2012) Ten years of pathway analysis: current approaches and outstanding challenges. PLoS Comput Biol 8: e1002375. https://doi.org/10.1371/journal.pcbi.1002375

# Chapter 6

# Feature Selection Applied to Microarray Data

**Amparo Alonso-Betanzos, Verónica Bolón-Canedo, Laura Morán-Fernández, and Borja Seijo-Pardo**

## Abstract

A typical characteristic of microarray data is that it has a very high number of features (in the order of thousands) while the number of examples is usually less than 100. In the context of microarray classification, this poses a challenge for machine learning methods, which can suffer overfitting and thus degradation in their performance. A common solution is to apply a dimensionality reduction technique before classification, to reduce the number of features. This chapter will be focused on one of the most famous dimensionality reduction techniques: feature selection. We will see how feature selection can help improve the classification accuracy in several microarray data scenarios.

**Key words** Microarray data, Dimensionality reduction, Feature selection

## 1 Introduction

Microarray technology is used to collect information from tissue and cell samples regarding gene expression differences that could be useful for diagnosing diseases. The classification of this type of data has been viewed as a particular challenge for machine learning researchers over the last 20 years, mainly due to their extremely high dimensionality (from 2000 to 25,000 features) in contrast with small sample sizes (often fewer than 100 patients). A typical classification task is to separate healthy patients from cancer patients based on their gene expression "profile" (binary approach). There are also microarray datasets in which the goal is to distinguish between different types of tumors (multiclass approach), making the task even more complicated.

Theoretically, it seems logical to expect that having more features should give more discriminant power to the algorithm. However, this is not the case for some induction algorithms, as many of them suffer from the problem of "curse of dimensionality." This phenomenon occurs when by increasing the number of features in a task, the training time required by the induction algorithms grows

Verónica Bolón-Canedo and Amparo Alonso-Betanzos (eds.), *Microarray Bioinformatics*, Methods in Molecular Biology, vol. 1986, https://doi.org/10.1007/978-1-4939-9442-7_6, 

exponentially. This scenario, apart from presenting high execution times, has other drawbacks as the appearance of false positives [1]. Besides, several studies have demonstrated that most of the genes measured in a DNA microarray experiment are not crucial to classification accuracy [2], as selecting a small number of discriminative genes ensures effective categorization of diseases [3–5]. For this reason, feature selection—defined as the process of identifying and removing irrelevant features from the training data—is receiving growing attention in gene selection for sample classification and is being increasingly used as preprocessing step in tackling microarray data. The selection of relevant genes while eliminating those which cannot add extra knowledge involves a number of important benefits such as improving the classification accuracy, helping biologists identify the underlying mechanisms relating gene expression to disease, or reducing the risk of overfitting. There are usually three varieties of feature selection methods: filters, wrappers, and embedded methods. While wrappers models involve optimizing a predictor as part of the selection process, filter models rely on the general characteristics of the training data to select features independent of any predictor. The embedded methods generally use machine learning models for classification, and then an optimal subset of features is built by the classifier algorithm. Traditionally, the most employed gene selection methods fall into the filter approach. Discretization as a step prior to feature selection has also received some degree of attention.

The remainder of this chapter is organized as follows. Subheading 2 states the foundations of feature selection and an experimental framework that can be used for future comparative studies. Then, Subheading 3 reviews the most up-to-date feature selection methods applied to microarray data. Subheading 4 presents several case studies in which (1) feature selection is combined with a previous discretization step, (2) the complexity of the datasets seems to be reduced after applying feature selection, (3) feature selection is applied in a distributed way, and (4) ensembles of feature selection are employed. Finally, Subheading 5 summarizes and closes the chapter.

## 2 Feature Selection

The advent of DNA microarray datasets has stimulated a new line of research both in bioinformatics and in machine learning. Especially for machine learning researchers, it poses a serious challenge because of having so many fields (in the order of thousands) relative to so few samples, which can lead to overfitting of the learning models. Moreover, several studies have shown that most genes measured in a DNA microarray experiment are not relevant in the accurate classification of different classes of the problem

[2]. To avoid the problem of the "curse of dimensionality" [6], dimensionality reduction techniques play a crucial role in DNA microarray analysis.

Dimensionality reduction techniques usually come in two flavors: *feature selection* and *feature extraction*. On the one hand, feature extraction techniques achieve dimensionality reduction by combining the original features. In this manner, they are able to generate a set of *new* features, which is usually more compact and of stronger discriminating power. It is the typical choice in applications such as image analysis, signal processing, or information retrieval. On the other hand, feature selection (FS) achieves dimensionality reduction by removing the irrelevant and redundant features. Due to the fact that feature selection maintains the original features, it is especially useful for applications where the original features are important for model interpreting and knowledge extraction, as it is the case in microarray analysis.

Feature selection methods are typically divided into three major approaches according to the relationship with the inductive learning method used to infer a model [7]:

- *Filters*, which rely on the general characteristics of training data and carry out the FS process as a preprocessing step with independence of the induction algorithm. This model is advantageous for its low computational cost and good generalization ability.
- *Wrappers*, which involve a learning algorithm as a black box and consists of using its prediction performance to assess the relative usefulness of subsets of variables. In other words, the FS algorithm uses the learning method as a subroutine with the computational burden that comes from calling the learning algorithm to evaluate each subset of features. However, this interaction with the classifier tends to give better performance results than filters.
- *Embedded methods*, which perform FS in the process of training and are usually specific to given learning machines. Therefore, the search for an optimal subset of features is built into the classifier construction and can be seen as a search in the combined space of feature subsets and hypotheses. In other words, ensemble methods learn which features best contribute to the accuracy of the model while the model is being created. This approach is able to capture dependencies at a lower computational cost than wrappers.

As can be seen above, each model has its advantages and disadvantages. In the case of microarray data, and due to the fact of having more features than samples, filters are usually preferred because of their low computational cost and low risk of overfitting.

In addition to this classification, feature selection methods may also be divided into univariate and multivariate types. Univariate methods consider each feature independently of other features, a drawback that can be overcome by multivariate techniques that incorporate feature dependencies to some degree, at the cost of demanding more computational resources. Besides, FS methods can be divided according to the output they produce: a subset of relevant features or an ordered ranking of *all* the features, according to their relevance. The first approach is known as *subset evaluation* and the latter as *individual evaluation* or *feature ranking*. Notice that, in the case of individual evaluation, it is necessary to establish a threshold in order to reduce the dimensionality of the problem.

### 2.1 Feature Selection Methods

As mentioned before, filter methods are the typical choice to achieve dimensionality reduction in microarray data. A brief description of some classical filter follows, which will be used in the experiments shown in Subheading 4.

- **Correlation-based feature selection** (CFS) is a simple multivariate filter algorithm that ranks feature subsets according to a correlation-based heuristic evaluation function [8]. The bias of the evaluation function is toward subsets that contain features that are highly correlated with the class and uncorrelated with each other. Irrelevant features should be ignored because they will have low correlation with the class. Redundant features should be screened out as they will be highly correlated with one or more of the remaining features. The acceptance of a feature will depend on the extent to which it predicts classes in areas of the instance space not already predicted by other features.
- The **fast correlation-based filter** (FCBF) method [9] is a multivariate algorithm that measures feature–class and feature–feature correlation. FCBF starts by selecting a set of features that is highly correlated with the class based on symmetrical uncertainty (SU), which is defined as the ratio between the information gain and the entropy of two features. Then, it applies three heuristics that remove the redundant features and keep the features that are more relevant to the class. FCBF was designed for high-dimensionality data and has been shown to be effective in removing both irrelevant and redundant features. However, it fails to take into consideration the interaction between features.
- The **INTERACT** algorithm [10] uses the same goodness measure as the FCBF filter, i.e., SU, but it also includes the consistency contribution, which is an indicator of how significantly the elimination of a feature will affect consistency. The algorithm consists of two major parts. In the first part, the features are ranked in descending order based on their SU values. In the

second part, features are evaluated one by one starting from the end of the ranked feature list. If the consistency contribution of a feature is less than an established threshold, the feature is removed, otherwise it is selected. The authors stated that this method can handle feature interaction, and efficiently selects relevant features.

- **Information gain** [11] is one of the most common attribute evaluation methods. This univariate filter provides an ordered ranking of all the features and then a threshold is required. In this work the threshold will be set up selecting the features which obtain a positive information gain value.
- **ReliefF** [12] is an extension of the original Relief algorithm [13]. The original Relief works by randomly sampling an instance from the data and then locating its nearest neighbor from the same and opposite class. The values of the attributes of the nearest neighbors are compared to the sampled instance and used to update the relevance scores for each attribute. The rationale is that a useful attribute should differentiate between instances from different classes and have the same value, for instances, from the same class. ReliefF adds the ability of dealing with multiclass problems and is also more robust and capable of dealing with incomplete and noisy data. This method may be applied in all situations, has low bias, includes interaction among features, and may capture local dependencies which other methods miss.
- The **mRMR** (minimum redundancy maximum relevance) method [14] selects features that have the highest relevance with the target class and are also minimally redundant, i.e., it selects features that are maximally dissimilar to each other. Both optimization criteria (maximum relevance and minimum redundancy) are based on mutual information.
- **SVM-RFE** (support vector machine based on recursive feature elimination) is probably the most famous embedded method, proposed by Guyon [15] to specifically deal with gene selection for cancer classification. This embedded method performs feature selection by iteratively training an SVM classifier with the current set of features and removing the least important feature indicated by the SVM.

### 2.2 Experimental Framework

The goal of performing feature selection on microarray data can be twofold: class prediction or biomarkers' identification. If the goal is class prediction, there is a demand for machine learning techniques such as supervised classification. However, if the objective is to find informative genes, the classification performance is ignored and the selected genes have to be individually evaluated. The experiments that will be presented in this subsection are focused on class prediction, which is an important reason to use feature selection methods

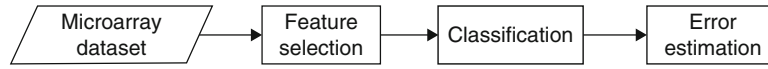


**Fig. 1** Microarray classification pipeline

**Table 1**
**Summary description of the binary datasets already divided into train and test**

| | | Train | | Test | |
|---|---|---|---|---|---|
| **Dataset** | **# Feats.** | **# Samp.** | **(%min,%maj)** | **# Samp.** | **(%min,%maj)** |
| Breast | 24,481 | 78 | (43.59, 56.41) | 19 | (36.84, 63.16) |
| Prostate | 12,600 | 102 | (49.02, 50.98) | 34 | (26.47, 73.53) |

**Table 2**
**Summary description of the binary datasets with only training set**

| **Dataset** | **# Feats.** | **# Samp.** | **(%min,%maj)** |
|---|---|---|---|
| Brain | 12,625 | 21 | (33.33, 66.67) |
| CNS | 7129 | 60 | (35.00, 65.00) |
| Colon | 2000 | 62 | (35.48, 64.52) |
| DLBCL | 4026 | 47 | (48.94, 51.06) |
| GLI | 22,283 | 85 | (30.59, 69.41) |
| Ovarian | 15,154 | 253 | (35.97, 64.03) |
| SMK | 19,993 | 187 | (48.13, 51.87) |

in microarray analysis. The typical microarray pipeline is formed by a feature selection step, followed by a classification stage which provides an error estimation, as seen in Fig. 1.

For this experimental study, we have considered nine widely used binary microarray datasets, which are available for download in [16, 17]. The reason for choosing binary datasets is that they are much more common in the literature than the multiclass ones. As a matter of fact, a typical microarray dataset consists of distinguishing between having a given cancer or not, therefore the great majority of the datasets are binary. Tables 1 and 2 summarize the properties of the selected datasets: for each dataset, the number of features (# Feats.), number of samples (# Samp.), and the percentage of examples of each class are shown. Datasets from Table 1 will be evaluated using a hold-out validation, while on those from Table 2 a fivefold cross-validation is performed.

**Table 3**
**Experimental results for C4.5 classifier on binary datasets after performing holdout validation**

| | | Breast | | | Prostate | | |
|---|---|---|---|---|---|---|---|
| | | **Ac** | **Se** | **Sp** | **Ac** | **Se** | **Sp** |
| No FS | | 0.74 | **1.00** | 0.58 | 0.26 | **1.00** | 0.00 |
| CFS | | 0.68 | 0.71 | 0.66 | 0.26 | **1.00** | 0.00 |
| FCBF | | 0.58 | 0.28 | 0.75 | 0.26 | **1.00** | 0.00 |
| INT | | **0.79** | 0.71 | 0.83 | 0.26 | **1.00** | 0.00 |
| IG | #10 | 0.47 | 0.28 | 0.58 | 0.26 | **1.00** | 0.00 |
| | #50 | 0.53 | 0.42 | 0.58 | 0.29 | **1.00** | 0.04 |
| ReliefF | #10 | 0.58 | 0.28 | 0.75 | 0.26 | **1.00** | 0.00 |
| | #50 | 0.42 | 0.71 | 0.25 | 0.29 | **1.00** | 0.04 |
| SVM-RFE | #10 | 0.58 | **1.00** | 0.33 | 0.32 | **1.00** | 0.08 |
| | #50 | 0.58 | **1.00** | 0.33 | 0.26 | **1.00** | 0.00 |
| mRMR | #10 | 0.58 | 0.71 | 0.50 | **0.41** | 0.88 | **0.24** |
| | #50 | 0.74 | 0.42 | **0.91** | 0.35 | **1.00** | 0.12 |

As for feature selection methods, we used the seven classical methods described in Subheading 2.1, widely used by the researchers in this field and all of them available in the well-known Weka tool [18], except for the mRMR filter, whose implementation is available for Matlab ®. Their performance may serve as a reference for the interested reader, so that comparative studies can easily be carried out based on this material. The three first feature selection methods (CFS, FCBF, and INTERACT) return a subset of features, while the remaining four (IG, ReliefF, mRMR, and SVM-RFE) provide an ordered ranking of the features. For the ranker methods, we show the performance when the top 10 and top 50 features are retained. After feature selection, three well-known classifiers were chosen to be applied: C4.5, Naive Bayes, and SVM (support vector machine). For more details, please refer to the work by Bolón-Canedo et al. [19].

#### *2.2.1 Holdout Validation Study*

This section reports the experimental results achieved over the binary datasets that were originally divided into training and test sets (*see* Table 1). Tables 3, 4, and 5 show the results achieved by C4.5, Naive Bayes, and SVM, respectively. These tables depict the classification accuracy (Ac), sensitivity (Se), and specificity (Sp) of the test datasets. For the sake of comparison, the first row shows those values without applying feature selection techniques. Notice that the C4.5 algorithm carries out a feature selection because not

**Table 4**
**Experimental results for Naive Bayes classifier on binary datasets after performing holdout validation**

| | | Breast | | | Prostate | | |
|---|---|---|---|---|---|---|---|
| | | **Ac** | **Se** | **Sp** | **Ac** | **Se** | **Sp** |
| No FS | | 0.37 | **1.00** | 0.00 | **0.26** | **1.00** | 0.00 |
| CFS | | 0.37 | **1.00** | 0.00 | **0.26** | **1.00** | 0.00 |
| FCBF | | 0.37 | **1.00** | 0.00 | **0.26** | **1.00** | 0.00 |
| INT | | 0.37 | **1.00** | 0.00 | **0.26** | **1.00** | 0.00 |
| IG | #10 | 0.32 | 0.85 | 0.00 | **0.26** | 0.88 | 0.04 |
| | #50 | 0.37 | **1.00** | 0.00 | 0.24 | 0.88 | 0.00 |
| ReliefF | #10 | 0.74 | 0.71 | 0.75 | 0.21 | 0.55 | **0.08** |
| | #50 | **0.89** | 0.85 | **0.91** | 0.21 | 0.77 | 0.00 |
| SVM-RFE | #10 | 0.68 | 0.85 | 0.58 | 0.18 | 0.55 | 0.04 |
| | #50 | 0.63 | **1.00** | 0.41 | **0.26** | **1.00** | 0.00 |
| mRMR | #10 | 0.37 | **1.00** | 0.00 | **0.26** | **1.00** | 0.00 |
| | #50 | 0.37 | **1.00** | 0.00 | **0.26** | **1.00** | 0.00 |

**Table 5**
**Experimental results for SVM classifier on binary datasets after performing holdout validation**

| | | Breast | | | Prostate | | |
|---|---|---|---|---|---|---|---|
| | | **Ac** | **Se** | **Sp** | **Ac** | **Se** | **Sp** |
| No FS | | 0.58 | 0.85 | 0.41 | 0.53 | **1.00** | 0.36 |
| CFS | | 0.68 | 0.85 | 0.58 | **0.97** | **1.00** | **0.96** |
| FCBF | | 0.58 | 0.28 | 0.75 | **0.97** | **1.00** | **0.96** |
| INT | | 0.74 | 0.71 | 0.75 | 0.71 | **1.00** | 0.60 |
| IG | #10 | 0.58 | 0.71 | 0.50 | **0.97** | **1.00** | **0.96** |
| | #50 | 0.79 | 0.57 | **0.91** | **0.97** | **1.00** | **0.96** |
| ReliefF | #10 | **0.84** | **1.00** | 0.75 | 0.94 | 0.88 | **0.96** |
| | #50 | **0.84** | 0.85 | 0.83 | **0.97** | **1.00** | **0.96** |
| SVM-RFE | #10 | 0.68 | **1.00** | 0.50 | 0.79 | **1.00** | 0.72 |
| | #50 | 0.68 | **1.00** | 0.50 | 0.74 | **1.00** | 0.64 |
| mRMR | #10 | 0.63 | 0.71 | 0.58 | 0.44 | **1.00** | 0.24 |
| | #50 | 0.68 | 0.71 | 0.66 | 0.91 | 0.77 | **0.96** |

all the attributes are considered when constructing the tree. The best results for the dataset and classifier are highlighted in bold.

Analyzing these tables, it is notable that the results obtained with SVM considerably outperformed those achieved by C4.5 or Naive Bayes. In fact, there is a clear tendency in the literature to use SVM as the standard de facto method to estimate performance measures and in González-Navarro [20], it is stated that the superiority of SVMs to other several classifiers seems to be beyond doubt. The Prostate dataset suffers from dataset shift [19], since the test dataset was extracted from a different experiment, and apparently C4.5 and Naive Bayes classifiers cannot solve this problem and opted for assigning all the examples to the majority class.

It is worth noting that the embedded method SVM-RFE, in spite of the fact that it is, in theory, better than the filters, achieves comparable or even worse results than them in terms of classification accuracy. Even when combined with the SVM classifier (*see* Table 5) it does not obtain the highest accuracy, contrary to what was expected. However, two datasets are not a sufficient basis from which to draw strong conclusions so it is necessary to check the results on the remaining datasets.

#### 2.2.2 Cross-Validation Study

Now we show the classification results obtained when applying the well-known cross-validation technique. To this end, a fivefold cross-validation was performed over the binary datasets presented in Table 2, which only have the training set available. Tables 6, 7, and 8 are devoted to the three classifiers employed (C4.5, Naive Bayes, and SVM), where 5 folds were considered and the results shown in the tables are the average test results for the 5 folds. These tables depict the classification accuracy (Ac), sensitivity (Se), and specificity (Sp). For the sake of comparison, the first row reports those values without applying feature selection. The best results for dataset and classifier regarding each measure are highlighted in bold.

Some interesting conclusions can be extracted by looking at the results reported in Tables 6, 7, and 8.

- The best performances are obtained by SVM and Naive Bayes classifiers. As mentioned above, some studies [20] noted the superiority of SVMs over other classifiers. On the other hand, the performance of C4.5 may be affected by its embedded feature selection, in some cases leading to an extremely reduced set of features which can degrade the classification accuracy.
- Focusing on the feature selection methods, on average for all datasets, the subset filters show an outstanding behavior, particularly CFS and INTERACT. It is surprising that SVM-RFE did not achieve the best results when combined with the SVM classifier, but the poor performance of the ranker methods can be explained by the restriction of having to establish a threshold

**Table 6**
**Experimental results for C4.5 classifier on binary datasets after performing fivefold cross-validation**

| | | | Brain | CNS | Colon | DLBCL | Gli85 | Ovarian | Smk | Avg |
|---|---|---|---|---|---|---|---|---|---|---|
| No FS | | Ac | **1.00** | 0.58 | 0.74 | 0.70 | 0.75 | 0.97 | 0.65 | 0.77 |
| | | Se | **1.00** | 0.64 | 0.60 | 0.69 | 0.81 | 0.95 | 0.66 | 0.77 |
| | | Sp | **1.00** | 0.48 | 0.82 | 0.70 | 0.63 | 0.98 | 0.62 | 0.75 |
| CFS | | Ac | **1.00** | 0.62 | 0.79 | 0.75 | 0.79 | 0.98 | 0.64 | **0.79** |
| | | Se | **1.00** | 0.64 | 0.68 | 0.78 | 0.81 | 0.95 | 0.56 | 0.78 |
| | | Sp | **1.00** | **0.58** | 0.85 | 0.71 | 0.75 | **0.99** | **0.71** | **0.80** |
| FCBF | | Ac | 0.86 | 0.48 | 0.79 | 0.73 | 0.82 | **0.99** | 0.61 | 0.75 |
| | | Se | 0.80 | 0.49 | 0.64 | 0.74 | 0.86 | **0.99** | 0.65 | 0.74 |
| | | Sp | 0.86 | 0.50 | 0.87 | 0.70 | 0.75 | **0.99** | 0.56 | 0.75 |
| INT | | Ac | **1.00** | 0.55 | 0.79 | 0.70 | 0.78 | 0.98 | 0.59 | 0.77 |
| | | Se | **1.00** | 0.54 | 0.72 | 0.74 | 0.81 | 0.98 | 0.51 | 0.76 |
| | | Sp | **1.00** | **0.58** | 0.82 | 0.66 | 0.71 | 0.98 | 0.66 | 0.77 |
| IG | #10 | Ac | 0.71 | 0.62 | 0.72 | 0.75 | **0.85** | 0.96 | 0.60 | 0.74 |
| | | Se | 0.70 | 0.69 | 0.78 | 0.79 | 0.88 | 0.93 | 0.71 | 0.78 |
| | | Sp | 0.70 | 0.48 | 0.70 | 0.71 | **0.79** | 0.97 | 0.48 | 0.69 |
| | #50 | Ac | 0.81 | 0.63 | **0.84** | 0.73 | 0.81 | 0.96 | 0.65 | 0.78 |
| | | Se | 0.70 | 0.67 | **0.83** | 0.69 | 0.86 | 0.96 | 0.62 | 0.76 |
| | | Sp | 0.87 | **0.58** | 0.85 | 0.74 | 0.71 | 0.97 | 0.67 | 0.77 |
| ReliefF | #10 | Ac | 0.72 | 0.47 | 0.72 | **0.85** | **0.85** | 0.97 | 0.65 | 0.75 |
| | | Se | 0.20 | 0.59 | 0.50 | 0.83 | 0.88 | 0.94 | **0.80** | 0.68 |
| | | Sp | **1.00** | 0.25 | 0.85 | **0.87** | 0.77 | **0.99** | 0.47 | 0.74 |
| | #50 | Ac | 0.62 | 0.53 | 0.82 | 0.73 | 0.82 | **0.99** | 0.61 | 0.73 |
| | | Se | 0.20 | 0.60 | 0.68 | 0.74 | 0.88 | **0.99** | 0.61 | 0.67 |
| | | Sp | 0.86 | 0.44 | **0.90** | 0.70 | 0.70 | **0.99** | 0.62 | 0.74 |
| SVM-RFE | #10 | Ac | 0.57 | **0.65** | 0.71 | 0.81 | 0.81 | 0.98 | 0.60 | 0.73 |
| | | Se | 0.00 | **0.74** | 0.60 | 0.82 | 0.85 | 0.98 | 0.65 | 0.66 |
| | | Sp | 0.87 | 0.48 | 0.77 | 0.79 | 0.75 | 0.98 | 0.55 | 0.74 |
| | #50 | Ac | 0.70 | 0.57 | 0.80 | 0.82 | 0.79 | 0.98 | 0.65 | 0.76 |
| | | Se | **1.00** | 0.61 | 0.77 | **0.84** | 0.83 | **0.99** | 0.62 | **0.81** |
| | | Sp | 0.56 | 0.49 | 0.82 | 0.79 | 0.70 | 0.98 | 0.66 | 0.72 |
| mRMR | #10 | Ac | 0.86 | 0.55 | 0.82 | 0.75 | 0.79 | 0.98 | **0.68** | 0.77 |
| | | Se | 0.90 | 0.72 | 0.68 | 0.79 | 0.86 | 0.96 | 0.71 | 0.80 |
| | | Sp | 0.87 | 0.23 | **0.90** | 0.70 | 0.61 | **0.99** | 0.64 | 0.70 |
| | #50 | Ac | 0.86 | 0.58 | 0.82 | 0.73 | 0.80 | 0.97 | 0.62 | 0.77 |
| | | Se | 0.90 | 0.70 | 0.77 | 0.69 | **0.91** | 0.96 | 0.66 | 0.80 |
| | | Sp | 0.87 | 0.39 | 0.85 | 0.74 | 0.54 | 0.98 | 0.57 | 0.71 |

for the number of features to retain. In the case of the subset filters, the number of features which form the final subset of features is the optimal one for a given dataset and method. However, the main disadvantage of rankers is the necessity of

**Table 7**
**Experimental results for Naive Bayes classifier on binary datasets after performing fivefold cross-validation**

| | | | Brain | CNS | Colon | DLBCL | Gli85 | Ovarian | Smk | Avg |
|---|---|---|---|---|---|---|---|---|---|---|
| No FS | | Ac | 0.67 | 0.60 | 0.55 | 0.92 | 0.84 | 0.93 | 0.63 | 0.73 |
| | | Se | 0.00 | 0.64 | 0.69 | **0.96** | 0.88 | 0.99 | 0.60 | 0.68 |
| | | Sp | **1.00** | 0.52 | 0.47 | 0.88 | 0.73 | 0.89 | 0.66 | 0.74 |
| CFS | | Ac | 0.81 | 0.67 | **0.85** | 0.90 | 0.82 | **1.00** | 0.65 | **0.81** |
| | | Se | 0.50 | 0.75 | 0.76 | **0.96** | **0.90** | 0.99 | 0.67 | 0.79 |
| | | Sp | **1.00** | 0.54 | **0.90** | 0.84 | 0.67 | **1.00** | 0.62 | 0.79 |
| FCBF | | Ac | 0.61 | **0.70** | 0.80 | 0.90 | 0.85 | 0.99 | 0.69 | 0.79 |
| | | Se | **1.00** | 0.77 | 0.76 | **0.96** | **0.90** | **1.00** | 0.72 | **0.87** |
| | | Sp | 0.40 | 0.58 | 0.82 | 0.84 | 0.74 | 0.99 | 0.65 | 0.72 |
| INT | | Ac | 0.81 | **0.70** | 0.77 | 0.90 | 0.82 | **1.00** | 0.64 | **0.81** |
| | | Se | 0.50 | 0.77 | 0.76 | **0.96** | 0.88 | **1.00** | 0.72 | 0.80 |
| | | Sp | **1.00** | 0.58 | 0.77 | 0.83 | 0.71 | 0.99 | 0.55 | 0.78 |
| IG | #10 | Ac | **0.86** | 0.63 | 0.79 | **0.94** | 0.85 | 0.96 | 0.61 | **0.81** |
| | | Se | 0.70 | 0.67 | 0.72 | **0.96** | 0.88 | 0.95 | 0.59 | 0.78 |
| | | Sp | 0.93 | 0.58 | 0.82 | **0.92** | 0.77 | 0.96 | 0.64 | 0.80 |
| | #50 | Ac | 0.81 | 0.63 | 0.77 | 0.92 | 0.85 | 0.98 | 0.66 | 0.80 |
| | | Se | 0.50 | 0.75 | 0.76 | **0.96** | 0.86 | 0.96 | 0.67 | 0.78 |
| | | Sp | **1.00** | 0.42 | 0.77 | 0.88 | 0.81 | 0.98 | 0.65 | 0.79 |
| ReliefF | #10 | Ac | 0.20 | 0.63 | 0.82 | **0.94** | 0.86 | 0.96 | 0.67 | 0.73 |
| | | Se | 0.20 | 0.72 | 0.72 | **0.96** | 0.88 | 0.95 | 0.71 | 0.73 |
| | | Sp | 0.20 | 0.48 | 0.87 | **0.92** | 0.81 | 0.96 | 0.63 | 0.70 |
| | #50 | Ac | 0.19 | 0.67 | 0.84 | 0.92 | **0.89** | 0.98 | 0.67 | 0.74 |
| | | Se | 0.50 | 0.72 | 0.77 | **0.96** | 0.86 | 0.95 | 0.72 | 0.78 |
| | | Sp | 0.07 | 0.58 | 0.87 | 0.88 | **0.97** | 0.99 | 0.61 | 0.71 |
| SVM-RFE | #10 | Ac | 0.62 | 0.68 | 0.73 | 0.92 | 0.82 | 0.99 | **0.71** | 0.78 |
| | | Se | 0.30 | 0.77 | 0.61 | 0.91 | 0.83 | **1.00** | **0.77** | 0.74 |
| | | Sp | 0.76 | 0.54 | 0.80 | **0.92** | 0.81 | 0.98 | 0.64 | 0.78 |
| | #50 | Ac | 0.67 | **0.70** | 0.76 | 0.92 | 0.88 | 0.99 | 0.70 | 0.80 |
| | | Se | 0.20 | **0.82** | 0.69 | 0.91 | 0.86 | **1.00** | 0.73 | 0.75 |
| | | Sp | 0.90 | 0.49 | 0.80 | **0.92** | 0.93 | 0.98 | 0.65 | **0.81** |
| mRMR | #10 | Ac | 0.73 | 0.63 | 0.80 | 0.92 | 0.85 | 0.99 | 0.67 | 0.80 |
| | | Se | 0.60 | 0.79 | 0.78 | **0.96** | 0.88 | 0.96 | 0.68 | 0.81 |
| | | Sp | 0.86 | 0.33 | 0.82 | 0.88 | 0.77 | **1.00** | 0.65 | 0.76 |
| | #50 | Ac | 0.63 | 0.62 | 0.80 | **0.94** | 0.80 | 0.99 | 0.67 | 0.78 |
| | | Se | 0.20 | 0.75 | **0.86** | **0.96** | 0.81 | 0.98 | 0.67 | 0.75 |
| | | Sp | 0.86 | 0.38 | 0.77 | **0.92** | 0.77 | 0.99 | **0.67** | 0.77 |

setting the threshold a priori. This has the risk of choosing too large or too small a number.

- All the methods tested except information gain are multivariate, which in theory should show a better performance than univariate methods. However, on average, for all the datasets evaluated,

**Table 8**
**Experimental results for SVM classifier on binary datasets after performing fivefold cross-validation**

| | | | Brain | CNS | Colon | DLBCL | Gli85 | Ovarian | Smk | Avg |
|---|---|---|---|---|---|---|---|---|---|---|
| No FS | | Ac | **0.68** | 0.67 | 0.77 | **0.96** | **0.92** | **1.00** | **0.72** | **0.82** |
| | | Se | 0.20 | 0.82 | 0.60 | 0.96 | **0.98** | **1.00** | 0.79 | 0.77 |
| | | Sp | 0.93 | 0.38 | 0.87 | **0.96** | 0.78 | **1.00** | 0.63 | **0.79** |
| CFS | | Ac | 0.61 | 0.62 | 0.81 | 0.88 | 0.88 | **1.00** | 0.64 | 0.78 |
| | | Se | 0.60 | 0.70 | 0.69 | 0.86 | 0.93 | **1.00** | 0.66 | 0.78 |
| | | Sp | 0.66 | 0.49 | 0.87 | 0.88 | 0.77 | **1.00** | 0.61 | 0.76 |
| FCBF | | Ac | 0.67 | 0.65 | 0.84 | 0.81 | 0.87 | **1.00** | 0.71 | 0.79 |
| | | Se | 0.00 | 0.80 | 0.73 | 0.82 | 0.93 | **1.00** | 0.76 | 0.72 |
| | | Sp | **1.00** | 0.39 | 0.90 | 0.79 | 0.73 | **1.00** | 0.64 | 0.78 |
| INT | | Ac | 0.61 | 0.62 | 0.81 | 0.88 | 0.88 | **1.00** | 0.66 | 0.78 |
| | | Se | 0.60 | 0.75 | 0.64 | 0.91 | 0.91 | **1.00** | 0.69 | 0.79 |
| | | Sp | 0.66 | 0.39 | 0.90 | 0.83 | 0.81 | **1.00** | 0.63 | 0.75 |
| IG | #10 | Ac | 0.48 | 0.63 | 0.81 | 0.94 | 0.91 | 0.98 | 0.64 | 0.77 |
| | | Se | 0.00 | 0.82 | 0.59 | 0.96 | **0.98** | 0.96 | 0.74 | 0.72 |
| | | Sp | 0.70 | 0.30 | **0.92** | 0.92 | 0.74 | 0.99 | 0.53 | 0.73 |
| | #50 | Ac | 0.66 | 0.67 | **0.85** | 0.94 | 0.86 | **1.00** | **0.72** | 0.81 |
| | | Se | **0.80** | 0.77 | **0.81** | 0.96 | 0.90 | **1.00** | 0.73 | **0.85** |
| | | Sp | 0.66 | 0.48 | 0.87 | 0.92 | 0.77 | 0.99 | 0.70 | 0.77 |
| ReliefF | #10 | Ac | 0.50 | 0.68 | 0.81 | 0.94 | 0.87 | 0.98 | 0.69 | 0.78 |
| | | Se | 0.00 | 0.87 | 0.60 | 0.96 | 0.96 | 0.94 | **0.82** | 0.74 |
| | | Sp | 0.73 | 0.34 | **0.92** | 0.92 | 0.66 | 0.99 | 0.54 | 0.73 |
| | #50 | Ac | 0.35 | **0.73** | **0.85** | 0.92 | 0.89 | **1.00** | 0.69 | 0.78 |
| | | Se | 0.00 | 0.82 | 0.72 | **1.00** | 0.93 | **1.00** | 0.74 | 0.74 |
| | | Sp | 0.53 | **0.58** | 0.92 | 0.84 | 0.82 | **1.00** | 0.64 | 0.76 |
| SVM-RFE | #10 | Ac | 0.62 | **0.73** | 0.73 | 0.87 | 0.86 | **1.00** | 0.70 | 0.79 |
| | | Se | 0.20 | 0.84 | 0.56 | 0.87 | 0.88 | **1.00** | 0.78 | 0.73 |
| | | Sp | 0.86 | 0.53 | 0.82 | 0.88 | 0.81 | **1.00** | 0.61 | **0.79** |
| | #50 | Ac | 0.48 | 0.72 | 0.71 | 0.88 | 0.89 | **1.00** | **0.72** | 0.77 |
| | | Se | 0.20 | 0.82 | 0.57 | 0.91 | 0.91 | **1.00** | 0.74 | 0.74 |
| | | Sp | 0.63 | 0.53 | 0.80 | 0.84 | **0.85** | **1.00** | **0.68** | 0.76 |
| mRMR | #10 | Ac | 0.53 | 0.65 | 0.77 | 0.92 | 0.89 | **1.00** | 0.68 | 0.78 |
| | | Se | 0.60 | **0.95** | 0.56 | 0.96 | 0.95 | 0.99 | 0.74 | 0.82 |
| | | Sp | 0.56 | 0.10 | 0.90 | 0.87 | 0.77 | **1.00** | 0.62 | 0.69 |
| | #50 | Ac | 0.49 | 0.70 | 0.84 | **0.96** | 0.89 | **1.00** | 0.68 | 0.79 |
| | | Se | 0.20 | 0.77 | 0.77 | **1.00** | 0.95 | **1.00** | 0.74 | 0.78 |
| | | Sp | 0.63 | 0.57 | 0.87 | 0.92 | 0.77 | **1.00** | 0.62 | 0.77 |

information gain obtains similar classification accuracy results to the other algorithms, considering that its complexity is lower.

- Some of the algorithms employed in this experimental study are based on information theory (information gain, mRMR, FCBF, and INTERACT) while the rest are based on correlation coefficients (CFS, ReliefF, and SVM-RFE). It seems that there is no

relation between the nature of the methods and their behavior. In fact, CFS achieves the highest accuracy on average for most of the classifiers, while ReliefF obtains in many cases the poorest results, and both of them are based on correlation coefficients.

## 3 Recent Approaches

DNA microarray technology [21, 22] allows for the monitoring and measurement of thousands of gene expression activation levels in a single experiment. Starting from 2001, release year of the human genome working draft [23], the field of bioinformatics has constantly grown and captured a remarkable interest of the scientific community, both in the area of computer science and in biology or medicine.

An interesting way to study the underlying biological processes is to apply a systematic and computational analysis on DNA microarray datasets. New machine learning methods to improve the analysis and understanding of these datasets have been developed in parallel [24]. In these scenarios, the analysis frequently involves class prediction, regression, feature selection, outliers detection, principal component analysis, discovering of gene relationships, and cluster analysis [25]. Due to their extremely high dimensionality, feature selection has a vital importance in the analysis of DNA microarray datasets [26].

Currently, new FS methods and improvements over existing ones continue to be developed, with the aim of dealing with DNA microarray dimensionality problem. One of the main research lines is the classification and diagnosis of cancer from DNA genes. In this sense, hybrid models are recently being proposed, integrating two different steps: a first step to perform a subset selection and a second in which the previous obtained subset is refined. Medjahed et al. [27] improved a classical *SVM-RFE* feature selection method by subsequently adding a binary dragonfly (BDF) metaheuristic [28]. Its efficiency was demonstrated by providing a higher classification accuracy rate in different microarray datasets often used in the literature. Jain et al. [29] integrated and improved correlation-based feature selection (CFS) with improved-binary particle swarm optimization (iBPSO), evaluating its performance on 11 benchmark microarray datasets of different cancer types. In a similar way, Alomari et al. [30] used a minimum redundancy maximum relevancy (MRMR) method together with bat-inspired algorithm (BA). BA is a recent swarm-based algorithm, which imitates the echolocation system of bat individuals.

Another recent approach is ensemble methodology, where several individual feature selection methods are combined with the main goal of reducing variability of individual methods, taking

advantage of their respective strengths and overcoming their weak points at the same time. Ebrahimpour et al. [31] executed an interesting study to discuss the rationale of ensemble feature selection. In addition, they proposed a new particular ensemble method called data perturbation strategy. This method consists in combining multiple selectors based on the same core algorithm but trained on different perturbed versions of the original data. This ensemble strategy was evaluated on ten public genomic datasets, providing useful conclusions and paving the way to design and develop new ensemble methods. Alkuhlani et al. [32] proposed a multistage feature selection approach combining three different filter methods to select the optimal subset of features on different DNA cancer datasets. Seijo-Pardo et al. [33] proposed a deep study on feature selection ensemble approaches, where several ranker methods are combined leading to different ensemble configurations based on different combination methods and threshold values. The performance of each ensemble configuration was tested on different scenarios, including DNA microarray datasets.

## 4 Case Studies

### 4.1 Discretization

Microarray datasets typically have a very large number of features and small number of instances. For this reason, machine learning algorithms in order to be effective need to confront the curse of dimensionality. One of the ways in which machine learning faces the problem is using feature selection, as many features are irrelevant or redundant for the classification task, a situation that is specially complicated if the training datasets are relatively small, as these irrelevancies/redundancies are harder to detect [34]. For this last reason, discretization is also an interesting method to apply to microarrays, as it greatly reduces the memory needed and improves classification accuracy. Discretization is a preprocessing step that aims at transforming quantitative data into qualitative data, as numerical attributes are converted to discrete or nominal attributes using a finite number of intervals, and thus obtaining a non-overlapping partition of the continuous input domain. Then, an association between each interval with a discrete value is established.

Most of the times, discretization is applied in a way transparent to the user, as several tools (as for example, Weka [18]) apply a discretizer by defect when using certain FS methods, mainly with many filters that can only work with discrete data [35]. Hence, a common practice to deal with numeric attributes is to discretize the data, mapping the real data into typically a small number of finite values, before conducting filtering. Furthermore, some of the features of microarray datasets have unbalanced values, and discretization seems to be a solution for this problem. There are

several advantages derived from the use of a preprocessing discretization [36]:

- The machine learning process will be more effective and efficient, as a reduced amount of data is needed in comparison with methods that use continuous values. Besides, as data is simplified, the learning process is usually faster, and the obtained results are more compact [7, 37, 38]. This characteristic is specially important in the present scenario of Big Data, for which discretization approaches are still scarce [39].
- The interpretability of the results is increased, as discrete values are easier to understand, use, and explain, in part because of the homogenization of the values carried out by the preprocessing [36].
- Additionally, by using discrete states part of the noise present in the original microarray data is absorbed, leading to a more robust behavior and better prediction accuracy [40, 41]

But not all are advantages, as discretization also implies a loss of information, and different discretization strategies might yield to distinct discrete-state models, even if the original data is the same. Thus, the selection of an appropriate discretization process has a major impact on the design and outcome of the inference algorithms, as there are a number of relevant issues that need to be considered [42, 43].

As stated above, microarray datasets are usually preprocessed using some feature selection algorithm that eliminates redundant and irrelevant genes. There are two general approaches commonly applied to feature (gene) selection: wrapper and filter approaches [7]. The wrapper approach uses a predetermined data mining algorithm and employs its performance as the evaluation criterion, while filters employ only general characteristics of the data to evaluate and select gene subsets. Wrappers tend to obtain better performances, but on the other hand they are very time consuming and have the risk of overfitting due to the small sample size of microarray data.

In the following, a summary of the results obtained by the authors in [44] are discussed. In the research described, several combination of discretizers and filters were tested before the classification step, with the aim of demonstrating (1) the improvement in accuracy with respect to using only the classifiers, and (2) that the combination of discretizer+filter achieves better performances than using FS alone. The discretization step is introduced both for aiding the filtering process and for dealing with the high number of genes with very unbalanced values present in microarray data.

The authors in [36] carried out an extensive study over an important number of different types of discretizers, classifiers, and dataset, aiming to give some recommendations on their combined

use; however, they are not focused on the gene selection problem. Conversely in Gallo et al. [42] there is an extensive review of the different state-of-the-art discretization methods, together with some aspects that need to be considered when designing or selecting a discretization approach for gene expression data. In [43], the authors carried out a comparative study using two commonly used discretizers (see description below), three filters (CFS, INTERACT, and Cons), and three classifiers (C4.5, Naive Bayes, and IB1). The filters and classifiers have been already described in other sections of this book, as in Subheading 2.1. The discretizers used are EMD (entropy minimization discretization) and PKID (proportional k-interval discretization). Both were selected due to their good results in a previous research [44]:

- EMD (entropy minimization discretization) [45]. This method evaluates as a candidate cut point the midpoint between each successive pair of the sorted values of an attribute. For evaluating each candidate cut point, the data are discretized into two intervals and the resulting class information entropy is calculated in polynomial time. A binary discretization is determined by selecting the cut point for which the entropy is minimal among all candidates. The binary discretization is applied recursively, always selecting the best cut point. A minimum description length (MDL) criterion is applied to decide when to stop discretization.
- PKID (proportional k-interval discretization) [46] is a method based on the idea that discretization bias and variance are related to interval size and interval number. This strategy seeks an appropriate trade-off between the bias and variance of the probability estimation by adjusting the number and size of intervals to the number of training instances. The following compromise is adopted: given a numeric attribute, with $n$ training instances with known values for it, it is discretized into $\sqrt{n}$ intervals, with $\sqrt{n}$ instances in each interval. Thus equal weight to both bias and variance management is given.

#### 4.1.1 Experimental Setting

The proposed method consists of applying a discretizer, a filter, and a classifier (as can be seen in Fig. 2).

Two different discretizers, three filters, and three classifiers were tested. The results obtained by all combinations over ten well-known microarray datasets are shown in Table 9. Some datasets were already originally divided into training and test sets; those that were not, and for which only training dataset was available (CNS, DLBCL, Colon, Lymphoma, Ovarian, and Breast) were randomly divided using the common rule 2/3 for training and 1/3 for testing.

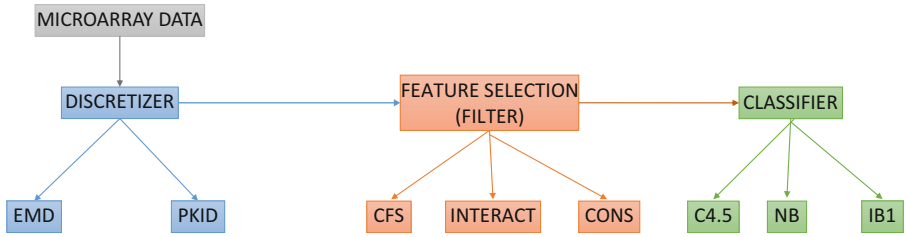


**Fig. 2** Steps and alternative methods tested

**Table 9**
**Dataset description**

| | | | Train | | Test | |
|---|---|---|---|---|---|---|
| **Dataset** | **Classes** | **Features** | **Samples** | **Distribution** | **Samples** | **Distribution** |
| Leukemia | 2 | 7129 | 38 | 71–29% | 34 | 59–41% |
| CNS | 2 | 7129 | 40 | 65–35% | 20 | 65–35% |
| DLBCL | 2 | 4026 | 32 | 50–50% | 15 | 53–47% |
| Colon | 2 | 2000 | 42 | 67–33% | 20 | 60–40% |
| Prostate | 2 | 12,600 | 102 | 49–51% | 34 | 26–74% |
| Lung | 2 | 12,533 | 32 | 50–50% | 149 | 90–10% |
| Ovarian | 2 | 15,154 | 169 | 35–65% | 84 | 38–62% |
| Breast | 2 | 24,481 | 78 | 56–44% | 19 | 37–63% |
| GCM | 14 | 16,063 | 144 | – | 46 | – |
| Lymphoma | 9 | 4026 | 64 | – | 32 | – |

Columns "Train distribution" and "Test distribution" show the percentages of the two classes for each binary dataset both in training and test sets

In Table 9, the distribution for training and testing datasets is shown in the corresponding columns for binary datasets. In the case of the multiclass, the distribution of GCM dataset is balanced and the percentages in the training and the test sets are roughly maintained. However, for lymphoma dataset, one of the nine classes corresponds with half of the samples and thus, there are some classes without representation in the training set or in the test set, making classification a much more complex task.

For carrying out the experiments, the Weka platform [18] was used, using default values for all parameters. For each dataset, a tenfold cross-validation was performed over the training set to derive a model that in turn will be applied to the test dataset, with the aim of checking the performance over a dataset with new data. In Table 10 it can be seen the results obtained after comparing the

performance achieved with a classifier (without preprocessing) and the best performance obtained by our proposed method grouped by dataset and classifier. Remember that 18 different methods were tested (Fig. 2), named by joining their acronyms by a "+" sign (for example, EMD+INT+NB). The table shows the validation accuracy (achieved using a tenfold cross-validation over the training dataset), the test accuracy (result of applying the model to the independent test dataset), and the number of genes required for each combination tested. The best test accuracy obtained for each dataset is emphasized in bold font.

As it can be seen in Table 10, for the majority of the datasets (the exceptions being colon and lymphoma), there is an

**Table 10**
**Best results obtained using the discretizer+feature selection filter+ classifier for each dataset**

| Dataset | Method | Validation accuracy | Test accuracy | No. of genes |
|---|---|---|---|---|
| Leukemia | C4.5 | 84.21 | 91.18 | 7129 |
| | PKID+Cons+C4.5 | 94.74 | **94.12** | **2** |
| | NB | 94.74 | 88.24 | 7129 |
| | PKID+CFS+NB | 100.00 | 94.12 | 18 |
| | IB1 | 89.47 | 70.59 | 7129 |
| | EMD+Cons+IB1 | 100.00 | 91.18 | 1 |
| CNS | C4.5 | 55.00 | 60.00 | 7129 |
| | EMD+INT+C4.5 | 82.50 | 65.00 | 47 |
| | NB | 65.00 | 60.00 | 7129 |
| | PKID+INT+NB | 90.00 | **75.00** | **4** |
| | IB1 | 50.00 | 55.00 | 7129 |
| | PKID+INT+IB1 | 85.00 | 65.00 | 4 |
| DLBCL | C4.5 | 87.50 | 86.67 | 4026 |
| | EMD+Cons+C4.5 | 96.88 | 86.67 | 2 |
| | NB | 84.38 | 93.33 | 4026 |
| | EMD+INT+NB | 100.00 | **93.33** | **36** |
| | IB1 | 71.88 | 73.33 | 4026 |
| | EMD+INT+IB1 | 96.88 | 66.67 | 36 |
| Colon | C4.5 | 83.33 | 90.00 | 2000 |
| | EMD+Cons+C4.5 | 97.62 | 85.00 | 3 |
| | NB | 54.14 | 70.00 | 2000 |
| | EMD+Cons+NB | 100.00 | 85.00 | 3 |
| | IB1 | 66.67 | **95.00** | **2000** |
| | EMD+Cons+IB1 | 97.62 | 85.00 | 3 |
| Prostate | C4.5 | 85.29 | 26.47 | 12,600 |
| | PKID+Cons+C4.5 | 88.24 | **73.53** | **2** |
| | NB | 63.73 | 26.47 | 12,600 |
| | PKID+Cons+NB | 85.29 | **73.53** | **2** |
| | IB1 | 84.31 | 52.94 | 12,600 |
| | PKID+Cons+IB1 | 88.24 | **73.53** | **2** |

(continued)

**Table 10**
**(continued)**

| Dataset | Method | Validation accuracy | Test accuracy | No. of genes |
|---|---|---|---|---|
| Lung | C4.5 | 71.88 | 81.88 | 12,533 |
| | EMD+Cons+C4.5 | 100.00 | 81.88 | 1 |
| | NB | 96.88 | 95.30 | 12,533 |
| | EMD+Cons+NB | 96.88 | 95.30 | 1 |
| | IB1 | 100.00 | 97.99 | 12,533 |
| | PKID+INT+IB1 | 100.00 | **100.00** | **40** |
| Ovarian | C4.5 | 91.72 | 98.81 | 15,154 |
| | EMD+Cons+C4.5 | 97.04 | 98.81 | 3 |
| | NB | 92.31 | 88.10 | 15,154 |
| | EMD+Cons+NB | 98.22 | **100.00** | **3** |
| | IB1 | 93.49 | 92.86 | 15,154 |
| | PKID+CFS+IB1 | 95.86 | **100.00** | 17 |
| Breast | C4.5 | 51.28 | 73.68 | 24,481 |
| | PKID+INT+C4.5 | 67.95 | **78.95** | **3** |
| | NB | 57.69 | 36.84 | 24,481 |
| | EMD+Cons+NB | 96.15 | 73.68 | 5 |
| | IB1 | 62.82 | 68.42 | 24,481 |
| | EMD+Cons+IB1 | 96.15 | 73.68 | 5 |
| GCM | C4.5 | 50.00 | 52.17 | 16,063 |
| | EMD+Cons+C4.5 | 61.81 | 41.30 | 9 |
| | NB | 67.36 | 52.17 | 16,063 |
| | EMD+Cons+NB | 68.06 | **54.35** | **9** |
| | IB1 | 58.33 | 45.65 | 16,063 |
| | PKID+CFS+IB1 | 86.11 | 52.17 | 1431 |
| Lymphoma | C4.5 | 70.31 | 75.00 | 4026 |
| | EMD+Cons+C4.5 | 85.94 | 59.38 | 3 |
| | NB | 65.63 | 68.75 | 4026 |
| | EMD+INT+NB | 98.44 | 81.25 | 160 |
| | IB1 | 95.31 | **87.50** | **4026** |
| | PKID+CFS+IB1 | 98.44 | 81.25 | 160 |

improvement in accuracy over the test set, with also a remarkable decrease in the number of features used. Focusing on the behavior of the approach regarding the accuracy of the classifiers, in the case of Naive Bayes and IB1, the proposed approach clearly improves the results of the classifier method alone, while with C4.5 it has a more irregular behavior, although in general (7 out of 10 datasets) the combination discretizer+filter+classifier maintains or improves the results of the classifier itself (for a more detailed explanation, please consult Bolón-Canedo et al. [43]). These results are in agreement with those findings in [36]. With respect to lazy and Bayesian learning, KNN, and Naive Bayes, for which the authors demonstrated that the subset of remarkable discretizers included PKID (remind that the study included several dataset types, not only microarrays).

The number of features employed, on the other hand, diminishes drastically for all classifiers and datasets, using features in the order of dozens of genes instead of thousands of them of the original dataset. This is an interesting aspect for enhancing transparency and explainability of the results, as it has the capacity to make it easier the identification of the underlying mechanisms relating gene expression to diseases. In [43] the authors also show the results obtained by their method in comparison with those of other authors that had used wrappers or other filters for classification of microarray datasets. Although the comparison should be interpreted in a broad sense, as the experimental conditions are slightly different, the conclusion is that the combination approach proposed (discretizer+filter+classifier) clearly outperforms them, with accuracy improvements slightly below 30%, which is notable in the case of wrappers than tend to be more precise in accuracy, although at the cost of higher computational complexities. Thus, the importance of using an adequate discretizer for microarray datasets has been established.

Some new tendencies in the area are related with the fact that generally discretization is applied before FS, as many of the latter methods require discrete data as input. Thus, and in order to be efficient, features are generally discretized individually. This univariate approach may not hold in cases where feature interactions exist, thus degrading the performance of the preprocessing stage, since as commented above, there is inevitably a loss of information in the process, and in this case information about feature interactions will probably be lost during the discretization process. Tran et al. [47] proposed a method that combines discretization and feature selection in just one step, using bare-bones particle swarm optimization (BBPSO). The method is named potential particle swarm optimization (PPSO), and uses a new representation aiming at reducing the search space of the problem, and also a new fitness function for evaluating candidate solutions to guide the search. Their results show a considerable reduction in the number of features selected, while performance regarding accuracy and generalization capabilities is maintained.

### 4.2 Complexity

Microarray data often presents some characteristics that can have a negative impact in the generalization ability of the classifiers. Some of the properties are data sparsity, a high number of features and a low number of samples, and class imbalance, since the cancer class tends to be rarer than the non-cancer class. Applying a proper feature selection method to the data can reduce the influence of those characteristics in the error rates of the classifiers induced. Lorena et al. [48] showed that, when a dimensionality reduction is accomplished, the impact of these characteristics tends to be decreased. Authors used a well-known feature selection procedure based on the Fisher linear discriminant analysis, due to its good

results in the context of microarray data [49]. Morán-Fernández et al. [50], making use of several data complexity measures, analyzed the intrinsic complexity of several microarray datasets with and without feature selection. Specifically, correlation-based feature selection and consistency-based filter were used in this work to reduce the microarray dimensionality. In both studies, data complexity measures proposed by Ho and Basu [51] were used. Data complexity analysis is aimed at representing data particularities that add complexity to classification tasks, such as overlaps between classes, separability, and decision boundary linearity. These measures are subsequently, as follows, grouped according to the aspect of the data they focus on.

1. Measures of overlap in feature values from different classes:
   - Maximum Fisher's discriminant ratio (F1).
   - Length of overlap region (F2).
   - Maximum (individual) feature efficiency (F3)
2. Measures of class separability:
   - Minimized sum of the error distance by linear programming (L1).
   - Error rate of the linear classifier by linear programming (L2).
   - Fraction of points on the class boundary (N1).
   - Ratio of average intra-/interclass NN distance (N2).
   - Error rate of the 1-NN classifier (N3).
3. Measures of geometry, topology, and density of manifolds:
   - Nonlinearity of a linear classifier by linear programming (L3).
   - Nonlinearity of the 1-NN classifier (N4).
   - Fraction of maximum covering spheres (T1).
   - Average number of points per dimension (T2).

From the previous cited works, it can be seen that feature selection seems to be effective in reducing microarray data complexity. Taking into account that higher values of F1 indicate simpler classification problems, in some cases the use of a feature selection method led to a decrease in the complexity of the classification problem. According to F2 values, there is less overlap between classes after applying feature selection, when compared to the original data. N1 and N2 values decreased significantly, indicating a simplification of the data structure. Higher N1 values indicate smaller separation in distributions and a more difficult classification task, so it seems logical to think that lower values after applying feature selection would improve classification

performance. N2 values for the original datasets were higher, suggesting that samples from the same class were dispersed in the feature space. However, values were lower after feature selection was applied. The values of the data complexity measures relative to linear separability, L1 and L2, increased significantly after applying feature selection, which might be because feature selection reduces the risk of overfitting.

### 4.3 Distributed Feature Selection

In the past, feature selection methods have been designed to run in centralized computing environments. But nowadays, several sources produce environments with distributed datasets where it is not legal or affordable to gather the data in a single location. Also big datasets might be collected in a central repository where processing imposes quite high computing requirements. In this context, most existing feature selection methods do not scale well, and their efficiency may significantly deteriorate to the point of becoming inapplicable. Thus, in order to improve the scalability of existing feature selection methods for high-dimensional datasets, feature selection is performed in distributed manner. Das et al. [52] developed a local distributed privacy preserving algorithm for feature selection in a large peer-to-peer environment. Banerjee et al. [53] proposed a secure distributed protocol that allowed feature selection for multiple parties without revealing their own data. Tan et al. [54] presented a new adaptive feature scaling framework for ultrahigh dimensional feature selection on big datasets. Other authors had considered parallel computing environments to re-implement feature selection algorithms. Peralta et al. [55] presented a feature selection algorithm based on evolutionary computation that used the MapReduce paradigm to obtain subsets of features for large datasets. Another proposal was a distributed parallel feature selection method that can read data in distributed form and perform parallel feature selection in symmetric multiprocessing mode via multi-threading and massively parallel processing [56].

Applied to microarray data, Bolón-Canedo et al. [57] proposed a method for distributing the feature selection process. The general idea consisted of dividing the microarray data into several nodes and then applying a feature selection method at each node performing several rounds to obtain a stable set of features. Later, a merging procedure is carried out to combine the partial results into a single subset of relevant features according to improvements in the classification accuracy. Data can be partitioned by samples (horizontally) or by features (vertically), depending on the characteristics of a particular problem. In the case of microarray data, the small sample size in contrast with the extremely high dimensionality demands the use of vertical partitioning. By applying the proposed distributed methodology to this domain, subsets will present a more balanced feature/sample ratio and, therefore, overfitting

**Table 11**
**Microarray datasets used in experiments, described in terms of the number of features, samples, and classes**

| Dataset | #Features | #Samples | #Classes |
|---|---|---|---|
| 11-Tumors 11 | 12,533 | 174 | 11 |
| Breast | 24,481 | 78 | 2 |
| CLL-SUB-111 | 111,340 | 111 | 3 |
| Gli85 | 22,283 | 85 | 2 |
| Lung cancer | 12,600 | 203 | 5 |

problems will be avoided. Although the experimental results over eight microarray datasets showed that the runtime was significantly reduced whereas classification performance was maintained—or even improving—compared to centralized algorithms, the merging procedure was dependent on the classifier used and in some cases it introduced an important computational burden. Thus, in order to overcome this drawback, Morán-Fernández et al. [58] designed a distributed method in which the final feature subset was updated according to the theoretical complexity of these features, by using the Ho and Basu data complexity measures [51] instead of the classification error. Hence, the new framework was not only independent of the classifier chosen, but also reduced considerably the computational time. The novel procedure was tested in five microarray datasets (Table 11) [59].

Figure 3 displays the classification accuracy obtained by the CFS filter and SVM classifier for the centralized approach (C) and distributed approaches (*D-F1*, *D-F2*, and *D-N2*) based on data complexity measure Fisher's multiple discriminant ratio (F1), length of the overlapping region (F2), and ratio of average intra-/interclass nearest neighbor distance (N2), respectively, for each microarray dataset. As can be seen, for some datasets the distributed versions even slightly outperform the standard centralized one. If we focus on the computational time (Table 12), we can see that it was drastically shortened by applying the distributed approach.

In light of these results, authors concluded that their distributed proposal performed successfully, since the running time was considerably reduced and the classification accuracy did not drop to inadmissible values. In fact, their approach was able to match and in some cases even improve the standard algorithms applied to the non-partitioned microarray datasets. As the runtime was small for the three distributed approaches, it could be interesting to use the data complexity measure with the best classification accuracy (F1).

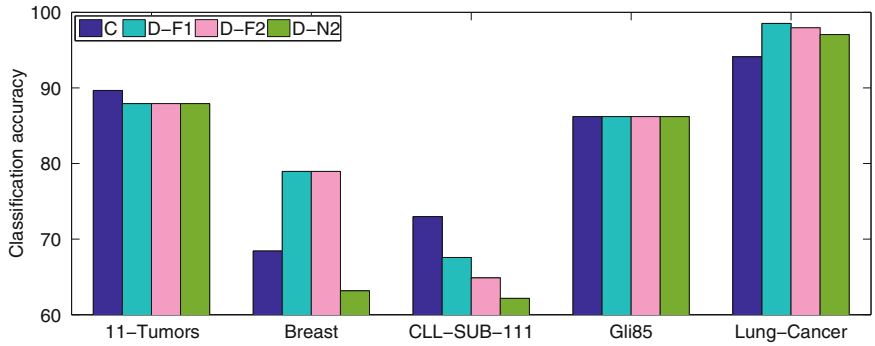


**Fig. 3** Classification accuracy obtained by the CFS filter and SVM classifier on the five microarray datasets

**Table 12**
**Runtime (seconds) for the CFS filter on the five microarray datasets**

| | | Distributed | | |
|---|---|---|---|---|
| | **Centralized** | ***D-F1*** | ***D-F2*** | ***D-N2*** |
| 11-Tumors | 7959.8 | 3.4 | 1.1 | 2.6 |
| Breast | 7669.2 | 3.6 | 1.3 | 2.8 |
| CLL-SUB-111 | 1335.0 | 3.4 | 1.1 | 2.6 |
| Gli85 | 7652.1 | 4.4 | 2.1 | 3.6 |
| Lung cancer | 9434.4 | 3.4 | 1.1 | 2.6 |

An open line of research is the inclusion of feature selection algorithms in parallel programming paradigms as Apache Hadoop [60] or Apache Spark [61]. Although both paradigms include machine learning libraries—Mahout and MLlib, respectively—with a number of learning algorithms such as SVM and Naive Bayes classification and k-means clustering, as yet, it includes no feature selection algorithms [62]. Some recent works, however, supply versions of well-known state-of-the-art algorithms, such as mRMR (minimum redundancy maximum relevance), ReliefF, and a framework of feature selection algorithms based on information theory [63–66].

### *4.4 Ensembles*

Machine learning methods have traditionally used a single learning model to solve a given problem. However, along the last few years, ensemble learning has become the focus of much attention based on the assumption that combining the output of multiple learning models in a particular problem improves the results obtained by a single model [67, 68]. This idea of ensemble learning has traditionally been applied to classification, although it can be also thought as

a means of improving other machine learning disciplines such as feature selection. In this field the goal is to select a subset of features that minimizes the prediction error of a given classifier [26]. Using a single feature selection method could generate local optima, but using an ensemble may be obtained more certain, precise, and accurate results. The aim of this approach is to achieve a method that, using ensembles and thus obtaining diversity in the selection, is able to reduce the variability of the individual methods, taking advantage of their respective strengths and overcoming their weak points at the same time. Another advantage of the ensemble method is the fact that the user is released from the task of choosing an adequate algorithm for each scenario, since this approach provides acceptable results independently of the characteristics of the data.

Improving the robustness of feature selection algorithms by using multiple feature selection evaluation criteria is proposed in recent works. For example, a recently ensemble-based multi-filter feature selection method [69] was proposed, combining the output of four filter methods to achieve an optimum selection for DDoS detection in cloud computing. Besides there are some other works in which all the feature selection methods of the final ensemble are ranker methods. Wang et al. performed a few outstanding works in this area, providing two interesting studies. One of them examines the ensembles of six commonly used filter-based rankers [70], while the other checks 17 different ensembles of feature ranking techniques [71], where the ensembles are composed of different numbers of rankers, ranging from 2 to 18 single feature selection methods. Focusing on microarray data processing, recent ensemble approaches can be found. One of this, min-max ensemble feature selection (M2-EFS) [72], is based on balanced data partition and min-max ensemble strategy. This approach can obtain higher performance than other classical ensemble methods especially for large-scale, high-dimensional, and imbalanced data. Another interesting study is the multicriterion fusion-based recursive feature elimination (MCF-RFE) algorithm [73], developed with the goal of improving both the classification performance and the stability of the feature selection results on microarray data.

In this field, it should be noted an exhaustive comparison study between different feature selection ensemble configurations [33]. This work builds ensembles based on six well-known individual ranker methods, differentiating two main steps:

- Combination step. Different rankings returned by each individual method are combining by an aggregation method.
- Thresholding step. Due to all FS methods employed are rankers, a threshold should be selected to obtain a practical subset of features. In this regard, fixed and automatic thresholds were used. Fixed thresholds retain a number of features according

**Table 13**
**Binary microarray datasets employed in the experimental study**

| Dataset | #Features | #Samples | | Train distribution (%) | Test distribution (%) |
|---|---|---|---|---|---|
| | | Train | Test | | |
| Colon | 2000 | 42 | 20 | 67–33 | 60–40 |
| DLBCL | 4026 | 32 | 15 | 50–50 | 53–47 |
| CNS | 7129 | 40 | 20 | 65–35 | 65–35 |
| Leukemia | 7129 | 38 | 34 | 71–29 | 59–41 |
| Lung | 12,533 | 32 | 149 | 50–50 | 90–10 |
| Prostate | 12,600 | 102 | 34 | 49–51 | 26–74 |
| Ovarian | 15,154 | 169 | 84 | 35–65 | 38–62 |

to dataset dimension, i.e., select a percentage like 10%, 25%, or $\log_2(n)$ features [74] where $n$ is the dataset dimension. Automatic thresholds retain a number of features according to dataset nature, i.e., thresholds based on complexity measures of overlap like F1, F2, and F3 (*see* Subheading 4.2).

Finally, after deriving a unique list of features, a classification step is carried out using five different classification methods. This approach was tested in seven DNA binary microarray datasets (available at http://datam.i2r.a-star.edu.sg/datasets/krbd/), which are detailed in Table 13.

Figure 4 displays the number of cases in which individual or ensemble approaches achieve the best classification error results. Ensemble approach was built using seven different combination methods (*SVM-Rank* [75], *Min* [76], *Median* [76], *Mean* [76], *GeomMean* [76], *Stuart* [77], *RRA* [78]). In addition, a thresholding step was applied to both individual and ensemble approaches using a fixed threshold ($\log_2(n)$) and an automatic threshold (F1 complexity measure). The performance of both approaches was tested by applying five different classification methods (*C4.5* [79], *Naive Bayes* [80], *K-nearest neighbor (k-NN)* [81], *random forest* [82], and *support vector machine (SVM)* [7]) on the aforementioned seven DNA microarray datasets (Table 13). This figure compares the traditional approach of applying individual feature selection methods (orange bars) and the novel approach of applying a feature selection ensemble method (blue bars). A total of 70 different experimentation cases (combination of 7 scenarios, 2 thresholds, and 5 classifiers) have been obtained. As can be seen, ensemble approach based on *Min* combination method has obtained the best performance. This ensemble achieved best results in 27 out of 70 experimentation cases, overcoming any of the individual

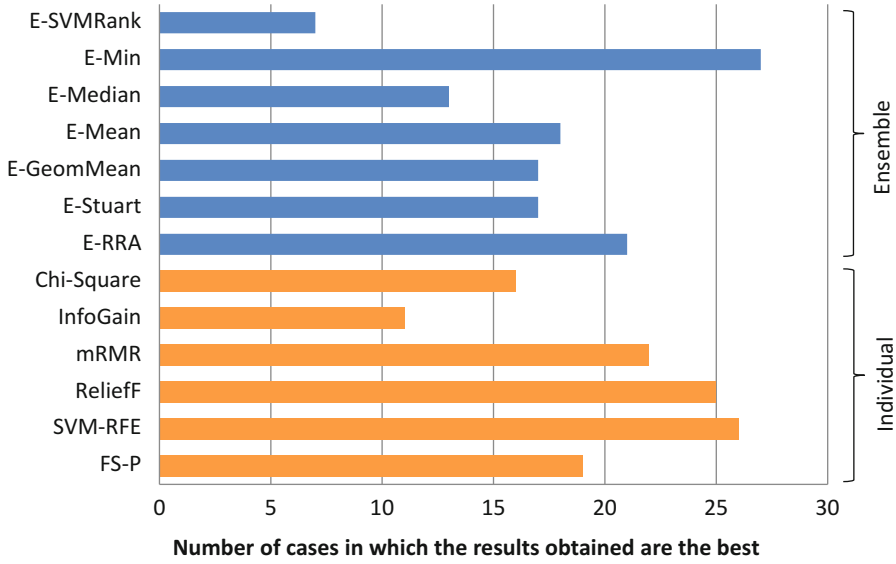


**Fig. 4** Number of cases in which individual or ensemble approaches obtain the best results. Blue bars denote different ensemble of feature selection methods (whose name begins with *E-*) that apply the *E-Combination-Method* to perform the ranking combination. Orange bars denote individual feature selection methods

methods. Despite of its simplicity, *Min* combination method within ensemble approach was able to get noteworthy results on DNA microarray scenarios.

## 5 Summary

DNA microarray data allows the expression levels of thousands of genes to be measured simultaneously. Such measurements can be used to assist with disease diagnosis, as they enable distinct kinds or subtypes of tumors to be classified according to expression patterns. In recent years, the classification of this type of data has been viewed as a particular challenge for machine learning researchers, mainly due to the mismatch between dimensionality and sample size. Several studies have demonstrated that most of the genes measured in a DNA microarray experiment do not actually contribute to efficient sample classification. In this sense, feature selection is advisable so as to identify the specific genes that enhance classification accuracy. In this chapter we have seen four different study cases: using a previous discretization step, carrying out a data complexity analysis, employing distributed approaches and finally, using ensemble approaches, for the study of DNA microarray datasets.

## References


1. Piatetsky-Shapiro G, Tamayo P (2003) Microarray data mining: facing the challenges. ACM SIGKDD Explor Newsl 5(2):1–5
2. Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri MA *et al* (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286(5439):531–537
3. Ding C, Peng H (2005) Minimum redundancy feature selection from microarray gene expression data. J Bioinform Comput Biol 3 (02):185–205
4. Wang Y, Tetko IV, Hall MA, Frank E, Facius A, Mayer KFX, Mewes HW (2005) Gene selection from microarray data for cancer classification–a machine learning approach. Comput Biol Chem 29(1):37–46
5. Xing EP, Jordan MI, Karp RM *et al* (2001) Feature selection for high-dimensional genomic microarray data. In: Proceedings of ICML, vol 1, pp 601–608. Citeseer
6. Jain A, Zongker D (1997) Feature selection: evaluation, application, and small sample performance. IEEE Trans Pattern Anal Mach Intell 19(2):153–158
7. Guyon I (2006) Feature extraction: foundations and applications, vol 207. Springer Science & Business Media, Berlin
8. Hall MA (1999) Correlation-based feature selection for machine learning. PhD thesis, Citeseer
9. Yu L, Liu H (2003) Feature selection for high-dimensional data: a fast correlation-based filter solution. In: Proceedings of the 20th international conference on machine learning (ICML-03), pp 856–863
10. Zhao Z, Liu H (2007) Searching for interacting features. In: Proceedings of the 20th international joint conference on artifical intelligence. Morgan Kaufmann Publishers Inc., San Francisco, pp 1156–1161
11. Hall MA, Smith LA (1998) Practical feature subset selection for machine learning. Comput Sci 98:181–191
12. Kononenko I (1994) Estimating attributes: analysis and extensions of relief. In: Machine learning: ECML-94. Springer, Berlin, pp 171–182
13. Kira K, Rendell LA (1992) The feature selection problem: traditional methods and a new algorithm. In: Proceedings of the National conference on artificial intelligence. Wiley, New York, pp 129–129
14. Peng H, Long F, Ding C (2005) Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans Pattern Anal Mach Intell 27(8):1226–1238
15. Guyon I, Weston J, Barnhill S, Vapnik V (2002) Gene selection for cancer classification using support vector machines. Mach Learn 46 (1–3):389–422
16. Feature Selection Datasets at Arizona State University (2018). http://featureselection.asu.edu/datasets.php. [Online; accessed Jan 2018]
17. Statnikov A, Aliferis CF, Tsamardinos I (2018) Gems: gene expression model selector. http://www.gems-system.org. [Online; accessed Jan 2018]
18. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. ACM SIGKDD Explor Newsl 11(1):10–18
19. Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A, Benítez JM, Herrera F (2014) A review of microarray datasets and applied feature selection methods. Inf Sci 282:111–135
20. González-Navarro FF (2011) Feature selection in cancer research: microarray gene expression and in vivo 1H-MRS domains. PhD thesis, Technical University of Catalonia
21. Dopazo J (2002) Microarray data processing and analysis. In: Methods of microarray data analysis II. Springer, Boston, pp 43–63
22. McConnell P, Johnson K, Lockhart DJ (2002) An introduction to DNA microarrays. In: Methods of microarray data analysis II. Springer, Boston, pp 9–21
23. International Human Genome Sequencing Consortium *et al* (2001) Initial sequencing and analysis of the human genome. Nature 409(6822):860
24. Lin SM, Johnson KF (2002) Methods of microarray data analysis: papers from CAMDA'00. Springer, New York
25. Brazma A, Vilo J (2000) Gene expression data analysis. FEBS lett 480(1):17–24
26. Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3(Mar):1157–1182
27. Medjahed SA, Saadi TA, Benyettou A, Ouali M (2017) Kernel-based learning and feature selection analysis for cancer diagnosis. Appl Soft Comput 51:39–48
28. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl 27 (4):1053–1073

29. Jain I, Jain VK, Jain R (2018) Correlation feature selection based improved-binary particle swarm optimization for gene selection and cancer classification. Appl Soft Comput 62:203–215
30. Alomari OA, Khader AT, Al-Betar MA, Abualigah LM (2017) Gene selection for cancer classification by combining minimum redundancy maximum relevancy and bat-inspired algorithm. Int J Data Min Bioinform 19 (1):32–51
31. Ebrahimpour MK, Eftekhari M (2017) Ensemble of feature selection methods: a hesitant fuzzy sets approach. Appl Soft Comput 50:300–312
32. Alkuhlani A, Nassef M, Farag I (2017) Multistage feature selection approach for high-dimensional cancer data. Soft Comput 21 (22):6895–6906
33. Seijo-Pardo B, Bolón-Canedo V, Alonso-Betanzos A (2017) Testing different ensemble configurations for feature selection. Neural Process Lett 46:1–24
34. Ferreira A, Figueiredo MAT (2011) Feature discretization and selection in microarray data. In: Proc international conf. on knowledge discovery and information retrieval - KDIR, pp 465–469
35. Liu H, Setiono R (1996) A probabilistic approach to feature selection - a filter solution. In: Proceedings of the 13th international conference on machine learning, pp 319–327
36. García S, Luengo J, Sáez JA, López V, Herrera F (2013) A survey of discretization techniques: taxonomy and empirical analysis in supervised learning. IEEE Trans Knowl Data Eng 25 (4):734–750
37. Cios KJ, Pedrycz W, Swiniarski RW, Kurgan L (2007) Data mining: a knowledge discovery approach. Springer, New York
38. Karlebach G, Shamir R (2008) Modelling and analysis of gene regulatory networks. Nat Rev Mol Cell Biol 9:770–780
39. Ramírez-Gallego S, García S, Mouriño-Talín H, Martínez-Rego D, Bolón-Canedo V, Alonso-Betanzos A (2016) Data discretization: taxonomy and big data challenge. WIREs Data Min Knowl Discovery 6(1):5–21
40. Gallo CA, Carballido JA, Ponzoni I (2011) Discovering time-lagged rules from microarray data using gene profile classifiers. BMC Bioinformatics 12:123
41. Ding C, Peng H (2005) Minimun redundancy feature selection from microarray gene expression data. J Bioinform Comput Biol 3:185–193
42. Gallo CA, Cecchini RL, Carballido JA, Micheletto S, Ponzoni I (2016) Discretization of gene expression data revised. Brief Bioinform 17(5):758–770
43. Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A (2010) On the effectiveness of discretization on gene selection of microarray data. In: Proc. 2010 international joint conference on neural networks, pp 3167–3174
44. Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A (2009) A combination of discretization and filter methods for improving classification performance in KDD Cup 99 dataset. In: Proc. 2009 international joint conference on neural networks, pp 359–366
45. Fayyad U, Irani K (1993) Multi-interval discretization of continuous-valued attributes for classification learning
46. Yang Y, Webb GI (2001) Proportional k-interval discretization for Naive-Bayes classifiers. In: Proceedings of the 12th international conference on machine learning, pp 564–575
47. Tran B, Xue B, Zhang M (2017) A new representation in pso for discretization-based feature selection. IEEE Trans Cybern 48:1733–1746
48. Lorena AC, Costa IG, Spolaôr N, De Souto MCP (2012) Analysis of complexity indices for classification problems: cancer gene expression data. Neurocomputing 75(1):33–42
49. Dudoit S, Fridlyand J, Speed TP (2002) Comparison of discrimination methods for the classification of tumors using gene expression data. J Am Stat Assoc 97(457):77–87
50. Morán-Fernández L, Bolón-Canedo V, Alonso-Betanzos A (2017) Can classification performance be predicted by complexity measures? A study using microarray data. Knowl Inf Syst 51(3):1067–1090
51. Ho TK, Basu M (2002) Complexity measures of supervised classification problems. IEEE Trans Pattern Anal Mach Intell 24(3):289–300
52. Das K, Bhaduri K, Kargupta H (2010) A local asynchronous distributed privacy preserving feature selection algorithm for large peer-to-peer networks. Knowl Inf Syst 24(3):341–367
53. Banerjee M, Chakravarty S (2011) Privacy preserving feature selection for distributed data using virtual dimension. In: Proceedings of the 20th ACM international conference on Information and knowledge management. ACM, New York, pp 2281–2284
54. Tan M, Tsang IW, Wang L (2014) Towards ultrahigh dimensional feature selection for big data. J Mach Learn Res 15:1371–1429
55. Peralta D, del Río S, Ramírez-Gallego S, Triguero I, Benitez JM, Herrera F (2015) Evolutionary feature selection for big data classification: a mapreduce approach. Math Probl Eng 2015:11pp.

56. Zhao Z, Zhang R, Cox J, Duling D, Sarle W (2013) Massively parallel feature selection: an approach based on variance preservation. Mach Learn 92(1):195–220
57. Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A (2015) Distributed feature selection: an application to microarray data classification. Appl Soft Comput 30:136–150
58. Morán-Fernández L, Bolón-Canedo V, Alonso-Betanzos A (2015) A time efficient approach for distributed feature selection partitioning by features. In: Conference of the Spanish Association for artificial intelligence. Springer, Cham, pp 245–254
59. Morán-Fernández L, Bolón-Canedo V, Alonso-Betanzos A (2017) Centralized vs. distributed feature selection methods based on data complexity measures. Knowl-Based Syst 117:27–45
60. Apache Hadoop (2018). http://hadoop.apache.org/. [Online; accessed Jan 2018]
61. Apache Spark (2018). https://spark.apache.org/. [Online; accessed Jan 2018]
62. Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A (2015) Recent advances and emerging challenges of feature selection in the context of big data. Knowl-Based Syst 86:33–45
63. Eiras-Franco C, Bolón-Canedo V, Ramos S, González-Domínguez J, Alonso-Betanzos A, Touriño J (2016) Multithreaded and spark parallelization of feature selection filters. J Comput Sci 17:609–619
64. Palma-Mendoza R-J, Rodríguez D, de Marcos L (2018) Distributed ReliefF-based feature selection in Spark. Knowl Inf Syst 57:1–20
65. Ramírez-Gallego S, Lastra I, Martínez-Rego D, Bolón-Canedo V, Benítez JM, Herrera F, Alonso-Betanzos A (2017) Fast-mrmr: fast minimum redundancy maximum relevance algorithm for high-dimensional big data. Int J Intell Syst 32(2):134–152
66. Ramírez-Gallego S, Mouriño-Talín H, Martínez-Rego D, Bolón-Canedo V, Benítez JM, Alonso-Betanzos A, Herrera F (2017) An information theory-based feature selection framework for big data under apache spark. IEEE Trans Syst Man Cybern Syst 48:1441–1453
67. Kuncheva LI (2004) Combining pattern classifiers: methods and algorithms. Wiley, New York
68. Kuncheva LI, Whitaker CJ (2003) Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Mach Learn 51(2):181–207
69. Osanaiye O, Cai H, Choo K-KR, Dehghantanha A, Xu Z, Dlodlo M (2016) Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. EURASIP J Wirel Commun Netw 2016 (1):130
70. Wang H, Khoshgoftaar TM, Gao K (2010) Ensemble feature selection technique for software quality classification. In: Proceedings of the SEKE, pp 215–220
71. Wang H, Khoshgoftaar TM, Napolitano A (2010) A comparative study of ensemble feature selection techniques for software defect prediction. In: 2010 ninth international conference on machine learning and applications (ICMLA). IEEE, Piscataway, pp 135–140
72. Ji W, Huang Y, Qiang B, Li Y (2017) Min-max ensemble feature selection. J Intell Fuzzy Syst 33(6):3441–3450
73. Yang F, Mao KZ (2011) Robust feature selection for microarray data based on multicriterion fusion. IEEE/ACM Trans Comput Biol Bioinform 8(4):1080–1092
74. Khoshgoftaar TM, Golawala M, Van Hulse J (2007) An empirical study of learning from imbalanced data using random forest. In: 19th IEEE international conference on tools with artificial intelligence, 2007, ICTAI 2007, vol 2. IEEE, Piscataway, pp 310–317
75. Joachims T (2002) Optimizing search engines using clickthrough data. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 133–142
76. Willett P (2013) Combination of similarity rankings using data fusion. J Chem Inf Model 53(1):1–10
77. Aerts S, Lambrechts D, Maity S, Van Loo P, Coessens B, De Smet F, Tranchevent L-C, De Moor B, Marynen P, Hassan B *et al* (2006) Gene prioritization through genomic data fusion. Nat Biotechnol 24(5):537–544
78. Kolde R, Laur S, Adler P, Vilo J (2012) Robust rank aggregation for gene list integration and meta-analysis. Bioinformatics 28(4):573–580
79. Quinlan JR (2014) C4. 5: programs for machine learning. Elsevier, Amsterdam
80. Rish I (2001) An empirical study of the Naive Bayes classifier. In: IJCAI 2001 workshop on empirical methods in artificial intelligence, vol 3. IBM, New York, pp 41–46
81. Aha DW, Kibler D, Albert MK (1991) Instance-based learning algorithms. Mach Learn 6(1):37–66
82. Breiman L (2001) Random forests. Mach Learn 45(1):5–32

# Chapter 7

# Cluster Analysis of Microarray Data

Manuel Franco and Juana-María Vivo

## Abstract

The cluster analysis has been widely applied by researchers from several scientific fields over the last decades. Advances in knowledge of biological phenomena have revived a great interest in cluster analysis due in part to the large amount of microarray data. Traditional clustering algorithms show, apart from the need of user-defined parameters, clear limitations to handle microarray data owing to its inherent characteristics: high-dimensional-low-sample-sized, highly redundant, and noisy. That has motivated the study of clustering algorithms tailored to the task of analyzing microarray data, which currently continue being developed and adapted. The present chapter is devoted to review clustering methods with different cluster analysis approaches in the challenging context of microarray data. Furthermore, the validation of the clustering results is briefly discussed by means of validity indexes used to assess the goodness of the number of clusters and the induced cluster assignments.

**Key words** Cluster analysis, Microarray data, High-dimensional-low-sample-sized, Clustering techniques, Multiclustering methods, Validity indexes, Cluster stability

## 1 Introduction

The task of assigning objects to clusters or classes from measurements made on these objects is a long-standing issue that has attracted the attention of research community in many scientific fields, including information retrieval, social sciences, machine learning, data mining, image segmentation, pattern recognition, and bioinformatics, among others. Based on the necessity to discover these clusters, when there is no a priori knowledge of them, it is found the so-called unsupervised methods, also known as cluster analysis, class discovery, or unsupervised pattern recognition. Alternatively, supervised methods are focused on understanding the basis for the classification into predefined classes from a set of labeled objects (learning set) in order to classify new unlabeled objects (prediction). They are also named classification, discriminant analysis, class prediction, or supervised pattern recognition, which are extensively used to find informative genes. Supervised methods can be applied when the phenotypes of the samples or

Verónica Bolón-Canedo and Amparo Alonso-Betanzos (eds.), *Microarray Bioinformatics*, Methods in Molecular Biology, vol. 1986, https://doi.org/10.1007/978-1-4939-9442-7_7, 

class patterns are known a priori. However, microarray data clustering is mainly performed by unsupervised or hybrid methods because expression patterns are unknown.

Since the 1990s, the burgeoning field of microarray data analysis has revived interest in cluster analysis by posing methodological and computational challenges [128] which enable to reveal the meaningful patterns hidden in this kind of data. This potential of clustering was proved in an early paper by Eisen et al. [38], who applied hierarchical clustering to identify functional groups of genes. Thus, at the beginning, the first-generation clustering algorithms were adopted by microarray users, but not all of them performed well to handle high-dimensional data because of the well-known curse of dimensionality [15]. In microarray data analysis, this problem is compounded by the relatively small sample size regarding the high dimensionality of the available data. Recently, sophisticated algorithms have been developed for performing efficient clustering on datasets with a large number of data, tackling some of the limitations of the traditional clustering methods as well as satisfying those specific requirements for microarray data [115].

By and large, a cluster analysis is a process which involves a number of consecutive interconnected stages: preprocessing, proximity calculation, clustering, and evaluation [65, 106]. Despite its undeniable fruitfulness in microarray research is increasingly visible in the literature (*see* Fig. 1), as regards second and third stages, yet there is no consensus about what is similar objects and how to

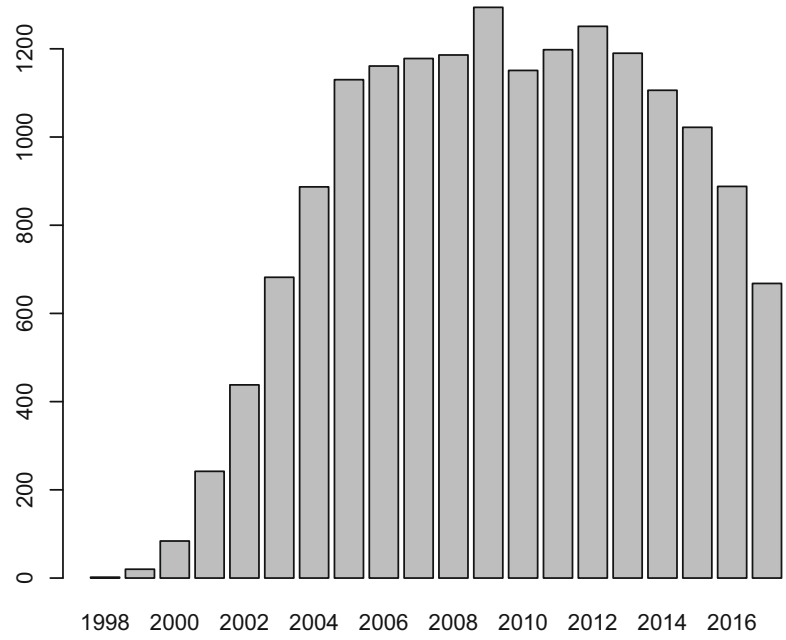


**Fig. 1** Results produced by a quick search from ISI-WoS in the period 1998–2017 and using the keyword microarray clustering

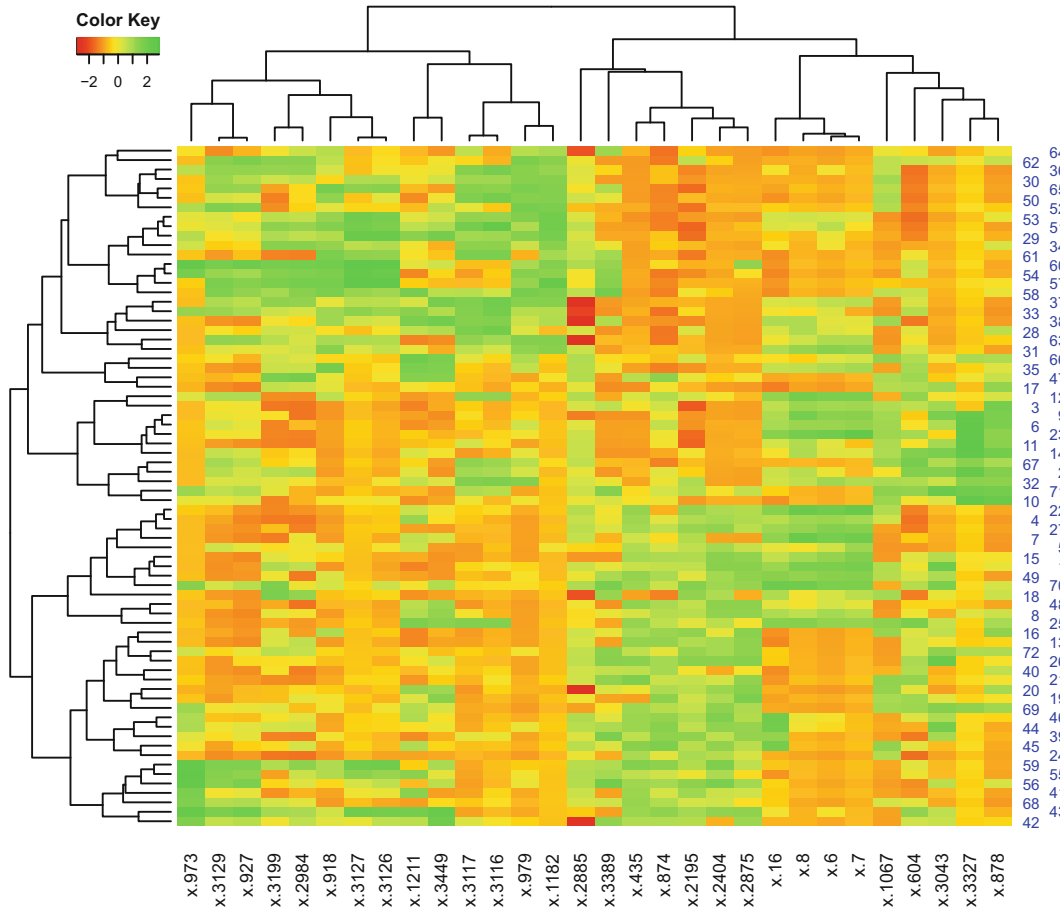


**Fig. 2** Heatmap from sample-based clustering and gene-based clustering of 30 genes chosen from the gene expression data of Golub et al. [50]

cluster such objects together. As well-known, the resulting outcomes of the unsupervised methods depend on the clustering algorithms as well as the distance metrics which are taken into account. However, as a response to both crucial points, many authors have frequently addressed their research on one of these issues.

Clustering has become an essential tool to generate new hypothesis from gene expression data, even more it has been usually deemed one of the first steps in gene expression analysis [32], and its large amount of applications has been commonly split into two categories: sample-based clustering and gene-based clustering [14, 33, 68] (*see* Fig. 2). The first one of them is focused on clustering biological samples to discover the phenotypic structures or substructures of samples, becoming usual in high throughput cancer studies since the seminal microarray analysis paper given by Golub et al. [50], in which it was demonstrated that the phenotypes

of samples can be separated considering only a small subset of informative genes whose expression levels are strongly correlated with the distinctions. Considering genes as objects and samples as features, gene-based clustering is aimed at identifying genes expressed differentially and groups of genes or clusters with similar expression patterns or profiles, and obtaining expression measurements in order to determine genes with similar function or co-regulated [38]. Currently, a third category, called subspace clustering, has been deemed, which is focused on finding clusters consisting of a subset of genes across a subset of samples that might have different feature spaces [69].

In the literature, numerous papers are devoted to review clustering approaches. Sneath and Sokal [117], Bezdek [19], Kaufman and Rousseeuw [74], Jain et al. [65], Aggarwal and Reddy [2], and Saxena et al. [111] provide good backgrounds on clustering methodologies. Sheng et al. [115] give an overview on the clustering algorithms of both the first and second generation which are selected by "their popularity, their ability to handle the specific characteristics of microarray data, and inevitably some personal biases." Other interesting reviews of clustering algorithms applied to microarray gene expression data analysis have been presented by Jiang et al. [69], Wong [130], Belacel et al. [14], and Kerr et al. [76], among others. A detailed and comprehensive review of traditional and modern clustering algorithms is given by Xu and Tian [132], introducing the core idea beyond them, and specifying their time complexity, weakness, and strengths. Aghabozorgi et al. [3] provide a review of time-series clusterings which tackles the clustering algorithms developed to time-series data during the last decade, some of which have been applied to microarray time-series data. The reader can also find an updated survey given by Oyelade et al. [99] on clustering approaches applicable to gene expression data.

Furthermore, commercial and non-commercial software tools are available to assist in clustering analysis in the challenging context of microarray data. Among the most widely used open-source software projects are found the statistical packages written in R project (https://www.r-project.org/), and those specifically designed for the genome data analysis through the Bioconductor (http://www.bioconductor.org). The growing and fast development of freely available packages has enabled statistical techniques to be implemented usually in the R environment, playing an important role in the advance of the scientific knowledge based on microarray data. Interested readers can see the "Cluster Analysis and Finite Mixture Models" CRAN task view at https://cran.r-project.org/web/views/Cluster.html for a list of packages dedicated to this field. In addition, Bioconductor clustering packages are enumerated at http://bioconductor.org/packages/release/BiocViews.html#___Clustering.

Owing to the increasing development of the algorithms for clustering gene expression data, it is not attempted to carry out an exhaustive review of the state-of-the-art. In this chapter, the authors have intended to include popular clustering algorithms, both traditional and modern, which have been used in the microarray analysis.

This chapter is organized as follows: Section 2 discusses the curse of the dimensionality in the context of microarray data. Furthermore, it is presented the preprocessing steps needed to be carried out before performing cluster analysis, being a prerequisite to mitigate the difficulties derived from the own nature of microarray data. Section 3 displays some distance metrics available for distance-based clustering algorithms; weakness, strengths, appropriateness, and differences are briefly indicated. Section 4 gives an overview of clustering algorithms, both the traditional and the modern algorithms, which have been used in microarray analysis, identifying some drawbacks of them in handling of the inherent characteristics of microarray data, and including new algorithms proposed in order to overcome such pitfalls. Section 5 provides a brief survey about biclustering and triclustering algorithms, and finally, Sect. 6 briefly discusses validity indexes to evaluate the goodness of the number of clusters and the stability of resulting cluster assignments.

## 2 Preprocessing Steps

The malediction of "the curse of dimensionality" is a term coined by Bellman [15] referred to as the soaring growth in the difficult posed by data analysis techniques as the number of dimension increases. For the sake of clarifying, Steinbach et al. [118] explain it simply by considering 100 points uniformly distributed over the unit interval [0, 1]. By supposing that such a unit interval is split into 10 subintervals, then it is highly probably that all bins have points. Now then, if these 100 points are distributed over the unit square fragmented into a lattice of 100 small square of side 0.1, it seems quite probable that some of their cells are empty. In addition, if the same amount of points is distributed in the unit cube which is divided into a three-dimensional lattice of 1000 subcubes of side 0.1, the majority of these small volumes are bound to be empty. As a result, the sparseness of data usually increases as the dimensionality of data increases.

Furthermore, clustering methods which work well in low dimension are affected by the curse of dimensionality. In microarray data analysis, the problem is magnified by the relatively small sample size regarding the high dimensionality of available datasets, close to 0.01 samples per dimension as highlighted by Jain et al.

[66] and Wang et al. [128]. Unfortunately, this fact poses a difficulty to traditional clustering algorithms.

Within clustering framework, distance metrics suffer when the dimensionality increases, becoming gradually meaningless because the spatial density of points decreases, and then all points tend to be equidistant which means that the nearest neighbors are not stable. Indeed, it is a proven fact that relative difference of the distances between closest and farthest data points from a randomly chosen point goes to zero when all features are identically and independently distributed [18]. As regards absolute difference, research confirmed that $L_1$ increases by dimensionality, $L_2$ remains relatively constant, and $L_p$ metric for $p \geq 3$ tends to be meaningless as dimensionality increases [62]. These metrics are defined in Sect. 3.

In order to manage the curse of dimensionality, there are a lot to be said for reducing dimensionality in microarray data as a prerequisite for performing a cluster analysis. To deal with, feature selection and feature extraction are two preprocessing steps that can be applied [65, 106]. Feature selection is the process focused on selecting the meaningful features, getting rid of noise according to an evaluation criterion, increasing accuracy, and comprehensibility of the results [134]. By comparison, feature selection on microarray is more difficult owing to its relative small sample size in relation to the large number of genes. What is more, some genes are strongly correlated, and therefore, one out of hundred genes is enough to depict the data [29]. Three general feature selection methods are distinguished: filter models, wrapper models, and combination of both [106]. One of the main advantages of this preprocessing step is that the original features are preserved. In high dimensionality, feature selection also plays an important role for helping to detect the underlying grouping tendency in a much lower-dimensional subspace [128]. Other available preprocessing step is feature extraction consisting of procedures which aim to reduce the dimensionality without losing relevant information by projecting objects from a higher dimensional space to a lower-dimensional one, usually removing noise. Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) are mainly used for dimensionality reduction. Nevertheless, feature selection and feature extraction procedures may be unsuitable when clusters are in different subspaces, which is automatically enabled by algorithms for clustering high-dimensional data [101, 115]. Some of them are discussed further in Sect. 4.

Another key preprocessing step is the standardization (or rescaling) in order to prevent the effect that the different scales can have on the similarity measures commonly used for microarray data which we will study later on. For instance, it can be about genes exhibiting the same relative behavior, but diverging absolute behavior [115]. Several standardization approaches have been

proposed, such as min–max normalization, autoscaling, and decimal scaling [106].

On the other hand, microarray data usually contain a high rate of missing values which poses a great difficulty for the application of the clustering procedures [21]. In order to solve this drawback, a straightforward solution proposed is to remove rows and columns with missing values which may vary from 1 to 10% affecting up to 95% of genes [27]. Apart from biased estimates, the huge loss of the information derived from this strategy has been conducted to use metrics that admit missing values, and alternatively, algorithms for microarray missing value imputation. In the literature, it is found a wide range of papers relative to missing value imputation for microarray data, e.g., *see* [6, 27, 82].

## 3 Distance Metrics

To start with, the relationship between objects to be clustered is represented by a symmetric squared matrix referred to as proximity matrix. Its elements are given in terms of degree of similarity (or dissimilarity) between all the objects by means of a distance, which are known as distance metrics (or distance measures). Thus, given two objects from the $n$-dimensional space $\mathbb{R}^n$, $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ (where $x_i$ and $y_i$ are named attributes) the distance between them is denoted by $d(\mathbf{x}, \mathbf{y})$ and should be symmetric and non-negative, and also satisfy the triangle inequality given by: $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{y}, \mathbf{z})$, for all $\mathbf{z}$. In contrast to model-based clustering methods discussed later on, the performance of distance-based clustering algorithms depends on distance metric. However, there is no universally accepted distance measure for clustering. Popular distance measures widely used are the following, when the data to be clustered are continuous, e.g., *see* [25, 46, 51, 68].

- *Minkowski distance*, also referred to as $L_p$-norm. Minkowski family is given by $d_{mink}(\mathbf{x}, \mathbf{y}) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}$, where $p \in \mathbb{R}$ such that $p \geq 1$.
- *Euclidean distance*, also referred to as $L_2$ norm, measures the geometric distance between two objects, considering the length of the straight line that connects those two objects. In spite of its frequent use in clustering, this well-known measure poses a problem when two data objects have no attribute values in common, because the distance between them may be smaller than any other pair of objects presenting the same attribute values [116]. As a Minkowski distance, Euclidean distance is sensitive to outliers and can be affected by differences in feature scales, i.e., the largest scaled feature would dominate the rest of them. For that reason, as commented before, the first is to

transform all features to have similar scales at the outset of data analysis, for example, by standardizing them [65]. This measure is defined as $d_e(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$, or equivalently, its squared measure can be used, *Squared Euclidean distance*, $d_{se}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n}(x_i - y_i)^2$. Alternatively, it can be modified by assigning increasingly greater weight on objects that are further apart.

- To solve the above-mentioned problem, *average distance* is provided as a modified Euclidean distance which enhances the results. This measure is given by $d_{ave}(\mathbf{x},\mathbf{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2}$.
- *Weighted Euclidean distance*, where features may be weighed differentially, is given by $d_{we}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n} w_i(x_i - y_i)^2}$, where a weight is allocated to each feature.
- *Manhattan distance*, also referred to as *city block distance* or *taxicab distance* or $L_1$ norm, is a particular case of Minkowski metric when $p = 1$ given by the sum of the absolute differences, i.e., $d_{man}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n}|x_i - y_i|$. Manhattan distance is more robust that Euclidean distance.
- *Chebyshev distance*, also referred to as $L_\infty$ or *Maximum metric*, may be appropriate in cases that require to define two objects as "different" if they are different on any one of the dimensions. This metric is robust with respect to outliers. The Chebyshev distance is computed as $d_{cheb}(\mathbf{x},\mathbf{y}) = \max_{1\le i\le n}|x_i - y_i|$.
- *Mahalanobis distance* was introduced by Mahalanobis in [91]. This data-driven measure is based on correlations between variables, and given by $d_{mah}(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})S^{-1}(\mathbf{x}-\mathbf{y})^t}$, where $S$ is the covariance matrix of the data which needs to be estimated additionally. Mahalanobis distance is advised when clusters are assumed to have elliptical shapes [48]. However, it can be expensive in terms of computation, and the time complexity is given by $O(3n)$ [116]. With uncorrelated components, $S$ is equal to identity matrix, and then Mahalanobis distance is reduced to Euclidean distance.
- *Canberra distance* is a scaled relative of Euclidean and Manhattan distances, whose range varies from 0 to 1. The Canberra distance is defined as $d_{can}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n}|x_i - y_i|/(|x_i| + |y_i|)$.
- Widely used in clustering microarray data [38], *Pearson sample correlation distance* is very useful when the shape of the expression vector is more important than its magnitude, for example, by calculating the similarity between the shapes of two gene patterns. This correlation-based distance is defined as

$$d_{cor}(\mathbf{x},\mathbf{y}) = 1 - r(\mathbf{x},\mathbf{y}) = 1 - \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(x_i - \overline{y})^2}}, \tag{1}$$

where $\overline{x}$ and $\overline{y}$ are the means for $\mathbf{x}$ and $\mathbf{y}$, respectively. As it can derive from Eq. 1, $d_{cor}$ is bounded in $[0, 2]$. Furthermore, its sensitivity to outliers can be improved by using *Jackknife correlation* [61]. In addition, variations on this distance are based on either replacing $\overline{x}$ and $\overline{y}$ with 0 (known as *Cosine correlation distance*) or considering the absolute value of correlation. On the other hand, the insensitivity of $d_{cor}$ to the amplitude of the expression vector may make its square useful to encompass repressions such as a form of coexpression [115].

- Since outliers affect correlation-based distances, robust measures such as *Spearman sample correlation distance* and *Kendall's $\tau$ sample correlation distance* are preferred [46], which are given by $d_{spear}(\mathbf{x},\mathbf{y}) = 1 - \frac{\sum_{i=1}^{n}(x'_i - \overline{x'})(y'_i - \overline{y'})}{\sqrt{\sum_{i=1}^{n}(x'_i - \overline{x'})^2 \sum_{i=1}^{n}(y'_i - \overline{y'})^2}}$ and $d_{tau}(\mathbf{x},\mathbf{y}) = 1 - \tau(\mathbf{x},\mathbf{y}) = 1 - \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} C_{x_{ij}} C_{y_{ij}}}{n(n-1)}$, respectively, where $x'_i = rank(x_i)$, $y'_i = rank(y_i)$, $C_{x_{ij}} = sign(x_i - x_j)$, and $C_{y_{ij}} = sign(y_i - y_j)$, [7].

Finally, it should be noticed that Euclidean and correlation distances are equivalent after standardization, more specifically: $d_e(\mathbf{x},\mathbf{y}) = \sqrt{2n(1 - r(\mathbf{x},\mathbf{y}))}$.

For the sake of understanding of the difference among the above distance metrics, an illustrative example showing pairwise distance between five gene expression patterns by four different similarity measures can be found in [33]. A study on the impact of chosen distance on microarray data analysis is carried out in [46]. Recently, Shirkhorshidi et al. [116] focused on the behavior of similarity measures on the results of distance-based clustering algorithms in high-dimensional datasets. Jaskowiak et al. [68] conducted a similar study on microarray data.

## 4 Clustering Techniques

According to Jardine et al. [67], there is no precise definition about what determines a cluster. For a clearer understanding, Fig. 3 illustrates three different ways to divide 22 points into clusters.

Traditionally, clustering algorithms broadly fall into two main groups: hierarchical and partitioning, as depicted in Fig. 4. The algorithms of the first group produce nested sequences of clusters arranged as a tree in which is considered (1) *a nodo* (cluster), the union of its children (subclusters) as well as any one of the leaf

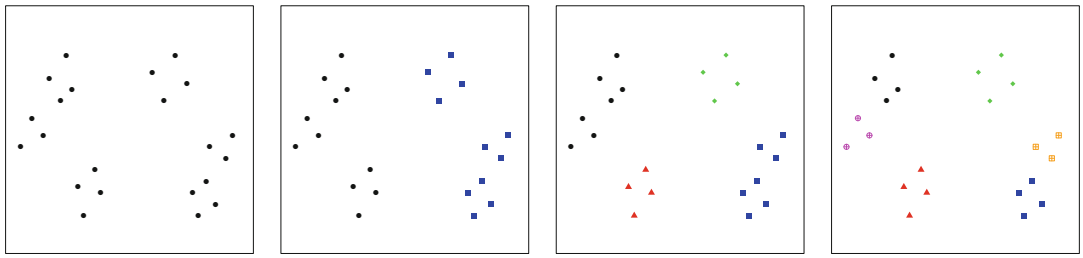

**Fig. 3** Different clusterings for a set of 22 points

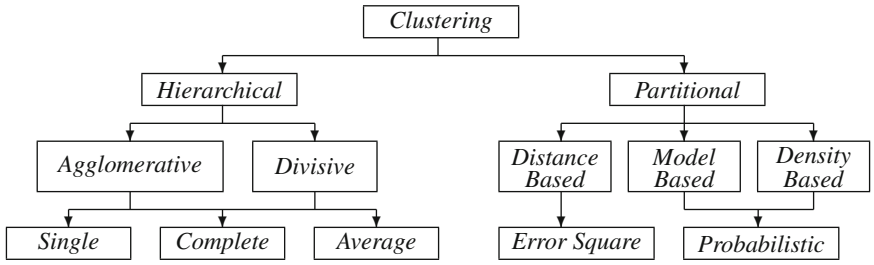


**Fig. 4** Taxonomy of clustering approaches [113]

nodes, and (2) *the root*, the cluster including all the objects, whereas the algorithms of the second group attempt to optimally separate $n$ objects into $k$ (non-nested or crisp) non-empty and non-overlapping subsets (clusters) whose union is the whole dataset. Nevertheless, some drawbacks have been reported using both hierarchical and non-hierarchical methods on microarray data, such as non-unique tree along with higher time and space complexity suffered from the first ones, and prefixed number of clusters required by the second ones. In order to overcome such limitations, a number of new clustering algorithms have been recently developed based on different concepts, e.g., *see* [99, 111].

Currently, as the term "cluster" encompasses different approaches with unsupervised methods, the categorization of clustering algorithms is not straightforward [17]. Taxonometric representations of clustering methods can be found in the literature [65, 99, 108], and for reader convenience, a close taxonomy of clustering approaches like the one discussed by Saxena et al. [111] is displayed in Fig. 4.

Different clustering approaches corresponding to above categories are explored below. Note that, apart from nested and non-nested clustering, other resulting types are mentioned throughout the present section: exclusive versus overlapping versus fuzzy, and complete versus partial, e.g., *see* [106, 122].

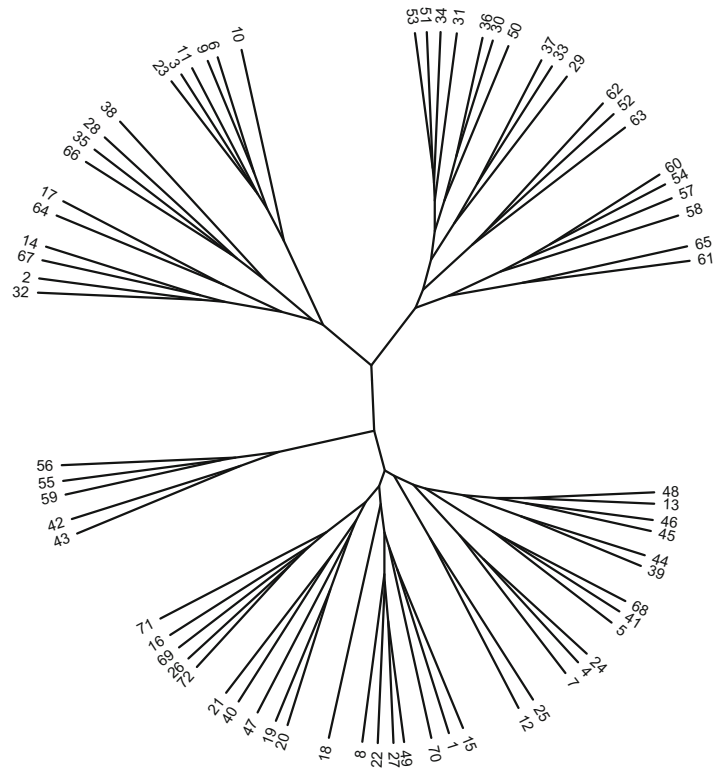


**Fig. 5** Unrooted phylogenetic tree dendrogram of 100 genes chosen from the gene expression data of Golub et al. [50]

### *4.1 Hierarchical Clustering*

Popularized by Eisen et al. [38] among others, hierarchical clustering has been extensively applied to analyze many types of microarray data (gene expression data, CGH arrays, and protein arrays). A hierarchical clustering is a sequence of partitions where each partition is nested into the next one, and a hierarchical clustering method [7, 23, 26, 38, 78, 89] is a procedure for transforming a proximity matrix into a sequence of nested partitions. The resulting hierarchical cluster structure is usually displayed by a dendrogram which is useful for interactive exploration and visualization (Fig. 5). By and large, it is considered two types of hierarchical clustering methods: agglomerative and divisive. The first one, also known as bottom-up approach, starts out with as clusters of size one as single objects. At each step, clusters merge into new clusters on the basis of their similarity, until all the objects are combined into a single cluster or a specified termination condition is reached. Nevertheless, the second algorithm, also called top-down approach, starts with a single cluster consisting of all the objects. At each step, resulting clusters are split until only clusters of single object remain

or termination condition set by user is reached. AGNES (AGglomerative NESting) and DIANA (DIvisive ANAlysis) are two earlier hierarchical clustering algorithms and the corresponding detailed implementations can be seen in Kaufman and Rousseeuw [74].

In order to define the inter-cluster distance, also the so-called cluster-to-cluster distance, different agglomerative hierarchical clusterings based on linkage metrics can be used in microarray data analysis.

- **Single-linkage clustering**, which is also known as the connectedness, the minimum method, or the nearest-neighbor method. The distance between two clusters is defined as the minimum distance from any object of one cluster to any object of the other one. As a tendency, clusters can be merged as long as two objects are close together regardless of other object contributions. Consequently, results in "chaining" may appear.
- **Complete-linkage clustering**, which is also called diameter, the maximum method, or the furthest-neighbor method. The distance between two clusters is calculated as the greatest distance between objects of the relevant clusters. Not surprisingly, this method tends to produce very compact clusters of elements and the clusters are often very similar in size (Fig. 6).
- **Average-linkage clustering**, which is also named as minimum variance method, is provided as an improvement in the face of

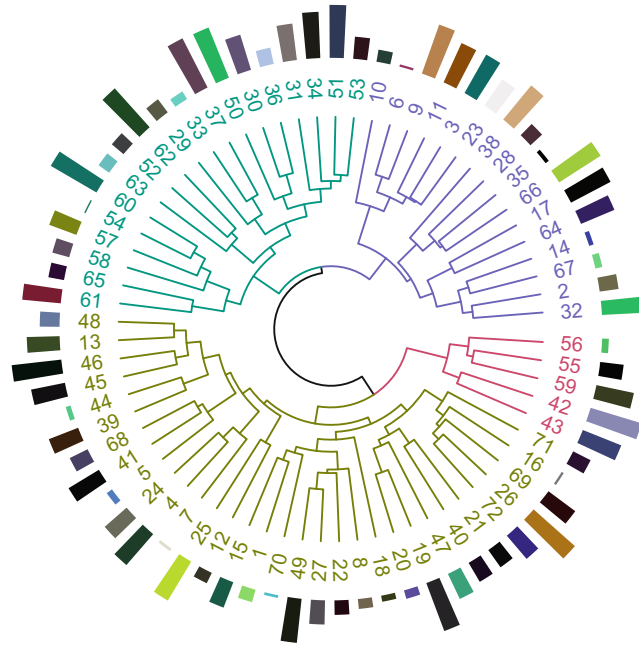


**Fig. 6** Circular dendrogram of 100 genes chosen from the gene expression data [50] by selecting complete-linkage clustering

the sensitivity to outliers suffered by single-linkage and complete-linkage clusterings. The distance between clusters is calculated using average values which can be obtained by different methods. For instance, the most common is the Unweighted Pair-Group Method Average (UPGMA). The average distance is computed from the distance between each object in a cluster and all other objects in another cluster. The two clusters with the lowest average distance are joined together to form a new cluster. Related methods use the centroids (centroid linkage or UPGMC) or the medians for the average.

- **Weighted pair-group average**. This method is similar to UPGMA, but the size of the respective clusters is used as a weight in the computations. This alternative method should be considered rather than UPGMA, when the cluster sizes are suspected to be greatly unequal. Sneath and Sokal [117] introduced the abbreviation WPGMA to refer to this method as Weighted Pair-Group Method using Arithmetic Averages. Related methods use the centroids (WPGMC) or the medians for the average.
- **Within-groups clustering**. This method is similar to UPGMA, but clusters are joined and a cluster average is used for further computations rather than the individual cluster objects, providing tighter clusters than UPGMA.
- **Ward's method**. This method presented by Ward [129] is based on the analysis of variance rather than considering distance or association measures. The objects are assigned to clusters by calculating the total sum of squared deviations from the mean of a cluster and joining clusters to produce the smallest possible increases in the sum of squared errors, i.e., to minimize the information loss. Note that a smaller sum of squared errors implies more similarity among the objects. More appropriate for quantitative variables, the clusters provided should be nearly elliptical-shaped.

Figure 7 displays dendrograms from three different linkage clusterings. Further, for an illustrative visual comparison, Fig. 8 shows two of them pointing out their differences.

As mentioned before, yet there is no theoretical rule for selecting the most appropriate method, but simulations may be a useful way to compare them [112]. The results provided by these linkage methods are slightly different as long as clusterings consist of compact and well separate clusters [34]. Apart from the linkage strategy, it should be pointed out that the resulting clustering depends on various parameters such as distance metric between objects as input to determine clusters. Likewise, it is worth mentioning that negatives associations are not considered, although the chosen distance measure supports them.

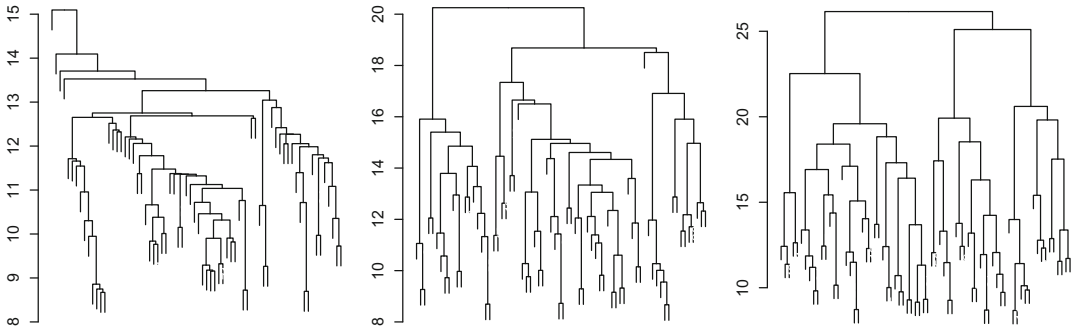

**Fig. 7** Hierarchical clustering of 100 genes chosen from the gene expression data [50], by selecting single-, average-, and complete-linkage clusterings, respectively

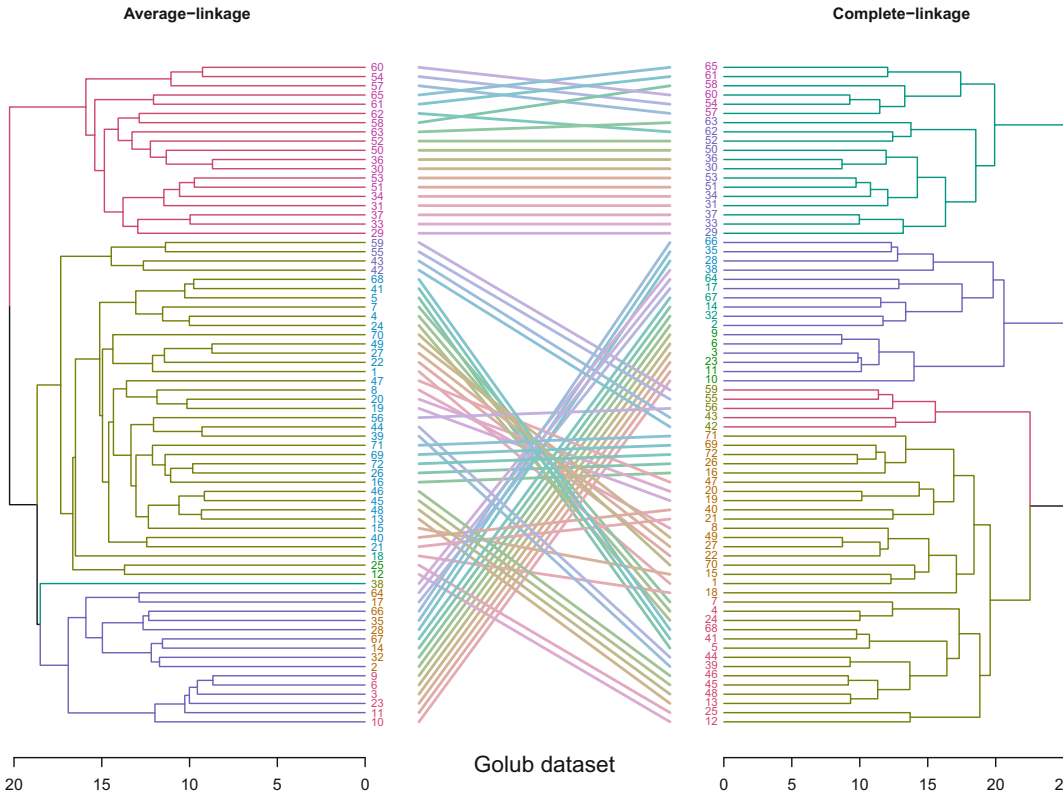


**Fig. 8** Tanglegram of 100 genes chosen from the gene expression data [50], for comparison between average- and complete-linkage clusterings

Hierarchical clustering algorithms require high-computational complexity. No less important, agglomerative and divisive traditional approaches suffer from local convergence due to its greedy nature for building dendrogram, since misclustering at the early steps is not reversible later, and then magnified as the procedure performs.

To address the issues of standard hierarchical clustering, recent algorithms have adopted one of the two following approaches. The first one represented by the algorithms such as CURE and CHAMELEON uses more complex principles to split or merge the clusters, and although both are irreversible yet, less errors are made because a better method is used. Clustering Using Representatives (CURE) is a hierarchical clustering algorithm which adopts a middle ground between centroid-based and all-point extremes [53]. CURE selects a number of well-scattered objects as representative of the cluster rather than mean, and then shrinking them toward the cluster centroid according to a determined fraction which dampens the adverse effect of outliers [99, 124]. In order to handle large datasets efficiently, CURE uses random sampling and partitioning to speed up clustering. Moreover, CURE is less sensitive to outliers and can identify both spherical and non-spherical clusters. CURE clustering algorithm was applied to gene expression by Guha et al. [53]. Developed by the authors of CURE, ROCK [52] deals with categorical datasets following a hierarchical agglomerative clustering. Iteratively, it merges clusters so as to try and maximize a criterion function, ROCK uses data sampling like CURE. CHAMELEON is a clustering algorithm that explores graph partitioning and dynamic modeling in hierarchical clustering to find the clusters in the dataset by using a two-phase algorithm [73]. Clusters are merged in the clustering process only if inter-connectivity and closeness between two clusters are high relative to the internal inter-connectivity and closeness within the clusters [111], thereby allowing it to choose the most similar pairs of clusters and get over the downside of incorrect merging decisions. However, CHAMELEON was not used to high dimensionality [110].

The second approach represented by algorithms like BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) intends to obtain an initial result by using a hierarchical agglomerative algorithm and then refining the outcome by using iterative relocation [130]. This approach uses the idea of Cluster Feature (CF) that is the triple ($n$, $LS$, $SS$), consisting of the number of objects, the linear sum of the feature values of the objects, and the sum of squares of the feature values of the objects stored in a CF tree form, i.e., a hierarchical data structure for multiphase clustering which reduces the volume of data. Two algorithms are used in BIRCH clustering: insertion and rebuilding [136]. The main advantages of BIRCH are the ability to deal with outliers and large datasets.

### 4.2 Partitioning Clustering

One of the most applied algorithms of partitioning clustering is $k$-means [90] which aims to find partitioning of the objects into a predefined number $k$ of non-empty and non-overlapping clusters, by maximizing both the compactness of the objects within cluster

and the separability between the clusters. In other words, minimizing the intra-cluster variance and maximizing the inter-cluster variance. A cluster centroid corresponds to the mean or median and the nearness is established by dissimilarity or similarity. Starting out with $k$ centroids which are chosen randomly, each object is assigned to the cluster with near centroid. At each interaction, the new $k$ centroids are recalculated from the previous partitioning, and then the new cluster assignment is carried out by reallocating each object to the nearest centroid, minimizing $\sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} \left\| \mathbf{x} - \mathbf{c}_j \right\|^2$, where $\mathbf{c}_j$ is the centroid of cluster $C_j$. The algorithm stops when objects do not change cluster or the centroids do not change. Furthermore, the complexity of $k$-means clustering method is linear, specifically $O(t\ k\ m\ n)$, where $t$ is the number iterations for convergence on a sample size of $m$ and $n$ features. However, the number of clusters must be specified by the user, and $k$-means clustering may become trapped in a local optimum because of this random initial clustering. In practice, the optimum number of clusters is obtained by using prediction-based resampling methods [35] or stability based methods instead of using a trial-and-error approach [115].

Alternatively, to overcome the sensitivity to noise and outliers presented by $k$-means clustering [74], some extents using the medoids rather than means can be applied. Medoid is the most centrally located object of the cluster whose average distance to all other objects in the same cluster is minimal. Unlike centroids, medoids can be always defined [64], for instance, in gene expression framework [125]. Even though medoid is an estimator of location more robust to outliers than mean, its calculation is computationally more costly since all pairwise distances are calculated within each cluster. It can be applied using Partitioning Around Medoids (PAM) algorithm [74] intended for spherical clusters, and its extension PAMSIL [125], which are popular $k$-medoid algorithms. Taking a dissimilarity matrix and a predefined number $k$ of clusters, whereas PAM is based on minimizing the sum of the dissimilarities of the objects to their nearest medoid, PAMSIL is focused on maximizing the average silhouette width of a cluster, whose mathematical expression will be introduced further in Sect. 6. According to [125], PAMSIL is good at finding small size clusters and the PAM medoids seem to be good starting values for PAMSIL. Unlike PAM, Clustering LARge Applications (CLARA) [74] and Clustering LARge Applications based upon RANdomized Search (CLARANS) [97] are two popular partitional clustering methods available for high-dimensional data. CLARA takes a sample of objects from the dataset on which PAM is applied to find medoids and returns its best clustering as output. However, CLARA depends on the sample, and the necessity to overcome such a disadvantage is fulfilled by CLARANS, whose core idea is random search to choose samples and combination of PAM and CLARA.

Observe that it should be run the algorithm using various random seeds in order to partition-optimization algorithms such as $k$-means and $k$-medoids.

### 4.3 Graph-Theoretic Clustering

Graph-theoretic clustering is an approach which depicts clusters in terms of a graph and considers the problem of clustering as finding the minimum cut or maximal cliques. From a dataset, a weighted graph or proximity graph, $G(V, E)$, can be constructed, in which each data object corresponds to a vertex or node and each pair of them is connected by an edge with a weight assigned reflecting its proximity. On the basis of a threshold value, proximity is mapped to either 0 or 1 with edges only existing between objects with proximity equal to 1.

Graph-theoretic algorithm based on Minimum Spanning Tree (MST) for clustering gene expression data [135] allows to transform multidimensional clustering problem into a tree partitioning problem since each cluster of the expression data corresponds to one subtree of the MST. Applied to gene expression data, Hartuv and Shamir [57] introduced a clustering algorithm HCS (Highly Connected Subgraph) based on graph connectivity. In this framework, a cut is defined as a set of edges whose removal results in a disconnected graph, and the edge connectivity of a graph $G(V, E)$ is determined by the minimum number of edges whose removal disconnects a graph. A cut is said a minimum cut if and only if its cardinal is equal to the edge connectivity of the graph. Furthermore, a graph or subgraph with more than a vertex is said highly connected if its edge connectivity is greater than half its total number of vertexes. Each highly connected subgraph is considered to be a cluster. HCS returns a minimum cut which recursively splits a graph $G(V, E)$ for finding a set of "highly connected" subgraphs (clusters).

In Ben-Dor et al. [16], Cluster Affinity Search Technique (CAST) was introduced to cluster gene expression data. In the graph, each node corresponds to a gene and each edge between two nodes is weighted according to the similarity of the two gene expression profiles. Inspired by the concept of a corrupted clique graph, this classic heuristic algorithm aims at searching for cliques, groups of closely connected nodes, in the graph [56]. CAST does not need a predefined number of clusters and can efficiently handle outliers. On the other hand, CAST requires a predefined parameter called affinity threshold, varying from 0 to 1, which controls the number of clusters.

Presented by Sharan and Shamir [113], CLuster Identification via Connectivity Kernels (CLICK) algorithm is an innovative graph-theoretical algorithm based on identifying kernels (cluster) of highly similar objects. CLICK makes the assumption that

pairwise similarity between objects is normally distributed. Thus, the weight of an edge is the probability that its vertexes belong to the same cluster. CLICK consists of two phases. First, the procedure iteratively finds the minimum cut in the graph and divides the dataset recursively into a group of connected components from the minimum cut. In the second phase, CLICK carries out two post-pruning steps to expand the kernels to final clustering. The adoption step repeatedly assigns remaining singletons to the kernels with the maximum similarity, and the merging step iteratively merges two clusters with similarity greater than a predefined threshold. This approach does not require initial number of clusters. Moreover, not only is CLICK scalable, but it is also very fast.

### 4.4 Evolutionary Approaches Based Clustering

Recently, Evolutionary Approaches (EA) [65] have been combined with clustering, as not only are they quite effective in searching optimal solutions, but they are also pretty time-consuming. Motivated by natural evolution, these approaches mainly include Evolution Strategies (ES), Evolutionary Programming (EP), Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO). For instance, GA are considered effective in dealing with NP-hard global optimization problems becoming suitable for overcoming clustering issues. However, the performance of GA is affected by predefined input parameters and inherent GA parameters fine-tuning for each single problem, e.g., general details of GA can be seen in [44] and the references given therein.

On the other hand, Krishna and Murty [79] introduced the Genetic $k$-means Algorithm (GKA) as a hybrid approach based on the genetic algorithm with a gradient descent algorithm and $k$-means algorithm. Inspired by GKA, Lu et al. [87] proposed a Fast Genetic $k$-means Algorithm (FGKA) where fitness function converges to the global optimum faster than previous algorithm. Moreover, some of these authors [86] proposed the Incremental Genetic $k$-means Algorithm (IGKA) based on clustering centroids incrementally. More recently, a GA-based Hierarchical Clustering method (HCGA), focused on finding global optima, provided by Castellanos-Garzón and Díaz [23] was applied to the analysis of DNA microarray data.

Other type of evolutionary approach is PSO developed by Kennedy and Eberhart [75] which is based on the simulation of social behavior of birds in a flock, and Xiao et al. [131] provided a hybrid clustering approach in order to improve the rate of convergence using SOM and PSO.

### 4.5 Grid-Based Clustering

These algorithms are based on a multiple-level granularity structure, i.e., a set of grid cells. Objects are assigned to the appropriate grid cell, and then the density of each cell is computed. Cells whose density is below a specified threshold are removed, whereas adjacent high-density cells are merged to form clusters [111]. The main

upside is its fast processing time [54] which depends on the number of grid cells [109]. Moreover, methods of adaptive grids can automatically regulate the size of grids based on the data distribution and do not necessarily need the user to specify the grid size or the density threshold by using STING, WaveCluster, CLIQUE, and MAFIA [4, 49, 114, 127].

STING (STatistical INformation Grid approach) by Wang et al. [127] is designed to provide "region oriented" required by using numerical attributes (spatial data).

WaveCluster [114] is a multi-resolution clustering approach, based on ideas of signal processing. This algorithm applies the wavelet transform to map spatial data preserving the relative distance between objects at different levels of resolution [17, 111]. Considered as both grid-based and density-based clustering approaches, Wavelet is scalable and can handle outliers.

CLIQUE (CLustering In QUEst) by Agrawal et al. [4] is a multiphase clustering algorithm which automatically finds subspaces of the highest dimensionality in a data space such that high-density clusters exist in those subspaces. It can be considered as both density-based and grid-based. MAFIA (Merging Adaptive Finite IntervAls) is a extent of CLIQUE algorithm which finds better quality clusters and achieves higher efficiency by non-uniform grid cells. These subspace clustering techniques focused on finding clusters within subspaces of the dataset can be useful for uncovering the complex relationships in microarray data [101].

### 4.6 Density-Based Clustering

This type of approaches is based on the common idea of defining clusters as connected dense components. The core idea behind density-based clustering is to find dense regions in the object space, regions with a high density. Density-based clusters are separated from other such regions by regions with low density, which means that for each object of a cluster, its neighborhood within a given radius $\varepsilon$ has to contain at least a minimum number (*minpts*) of other objects. Consequently, regions of lower object density are not assigned to any cluster. Such methods make assumptions neither about a specific shape for the cluster, nor about the number of cluster, nor about the distribution.

The algorithm Density-Based Spatial Clustering of Applications with Noise (DBSCAN) by Ester et al. [40] is based on the concepts of density and connectivity in respect of local distribution of nearest neighbors, and requires an $\varepsilon$-radius and a threshold *minpts* to be specified by user in advance. Moreover, DBSCAN cannot find meaningful clusters in data varying density. To overcome shortcomings of DBSCAN, the Ordering Points To Identify the Clustering Structure (OPTICS) algorithm is provided by Ankerst et al. [9]. By keeping the same two parameters, $\varepsilon$-radius and *minpts*, OPTICS covers a range of all variety $\varepsilon' \leq \varepsilon$. Instead of

relying only on the two parameters, OPTICS also stores two values for each object: the core-distance and reachability-distance. However, DBSCAN and OPTICS are not suitable for high-dimensionality data.

In Hinneburg and Keim [62] another density-based clustering algorithm, DENCLUE, is introduced to find arbitrary shaped clusters in high-dimensionality data by using a kernel density estimator. DENCLUE is focused on modeling the overall object density as the sum of influence functions of the data objects, where the influence functions describe the impact of a data object within its neighborhood. A cluster is defined by a local maximum of the estimated density function, called density attractor. Each object is associated with the density attractor that is in the direction of maximum increase in density from the corresponding object. It should be to notice that the quality of the resulting clustering depends on the selection of the two parameters, the one which sets the influence of an object in its neighborhood and the other one that establishes whether a density attractor is significant.

### 4.7 Model-Based Clustering

The core idea behind model-based clustering is to consider data as from a mixture distribution, e.g., see the pioneering works done by Banfield and Raftery [12] and Fraley and Raftery [43] for further details. The main purpose is to find the unobserved label for each object $x_i$ with $i = 1, 2, \ldots, n$, such as $z_i = g$ if $x_i$ comes from cluster $g$ with $g = 1, 2, \ldots k$, and $\{(x_i, z_i)\}_{i=1}^{n}$ is referred to as the complete dataset. In other words, objects are grouped by using finite mixture models in such way that each mixture component represents a cluster. Thus, it is said that a random vector $\mathbf{X}$ arises from a $k$-component (or $k$-cluster) mixture model if, for all $\mathbf{x} \in \mathbf{X}$, its density probability function can be written as $f(\mathbf{x}|\theta) = \sum_{g=1}^{k} \pi_g f_g(\mathbf{x}|\theta_g)$, where the mixing proportions $\pi_g \in [0, 1]$ such as $\sum_{g=1}^{k} \pi_g = 1$ and $f_g(\mathbf{x}|\theta_g)$ is the conditional density function of the $g$th mixture component for $g = 1, \ldots, k$ [93, 94].

In general for continuous data, and in particular for microarray, finite normal mixture distributions are commonly used [100]. Mixture distributions are fitted by maximum likelihood using the Expectation-Maximization (EM) algorithm [31]. This two-step iterative optimization algorithm is focused to find mixture model parameters that maximize the log-likelihood. These methods present upsides over heuristic approaches, for example, to asses uncertainty about resulting clustering and estimate the number of clusters [28]. Model-based clustering approaches for gene expression data are discussed in [69, 92], among others. A model-based clustering approach is AutoClass [1], where the finite mixture model is supplemented by a Bayesian method and the optimal is determined by EM algorithm. AutoClass is used to find class description, without specifying number of classes, and applied on microarray data.

Apart from this probabilistic approach, there are model-based clustering algorithms based on statistical learning method and neural network learning method [132]. Proposed by Kohonen [77], Self-Organizing Feature Map (SOM or SOFM) is a well-known model-based clustering approach [111] widely applied in gene expression clustering [14, 76], and based on a single-layered Artificial Neural Network (ANN) where the SOM is constructed by training [14]. From data objects as input, the output neurons represent clusters, but also objects connected to the prototype neurons. SOM generates an intuitive appealing map of a high-dimensional dataset in one-, two-, or three-dimensional space where similar clusters are placed close to each other. Before clustering starts, each neuron of the neural network is associated with a random weight pattern (also called reference vector). During training, objects are mapped repeatedly. Thus, clustering is performed by having neurons compete for the current training data object. The winning neuron is that one with the closest reference vector, which is updated to even closer. When the training is complete, the mapping of all objects to the output neurons conducts the identification of clusters.

SOMs are pretty used as vector quantization method [111]. SOMs are robust to noise and outliers as neuron learning process, dependent on distance metric and neighborhood function used [69, 76]. Due to their linear run time [14], SOMs have efficient in handling large-scale datasets. Tamayo et al. [121] depicted one of the first uses of SOMs for clustering of microarrays.

To get over pitfalls of SOM, Su and Chang [120] developed the Double SOM (DSOM) in order to determine the number of clusters. In this approach, each node is associated with an $n$-dimensional weight vector and a two-dimensional position vector which are updated as learning process goes. Even though the number of clusters can be revealed from the final location of position vectors, ADaptive DSOM (ADSOM) is an improvement on DSOM by updating the free parameters involved in DSOM [33].

Applied on microarray data, Hsu et al. [63] combined dynamic SOM tree and Growing SOM (GSOM) getting an appropriate number of clusters and optimal. Self-Organizing Tree Algorithm (SOTA) [60], Dynamically Growing Self-Organizing Tree (DGSOT) algorithm [88], and Growing Hierarchical Tree SOM (GHTSOM) [42] are based on neural networks combined with hierarchical clustering. Such tree structured self-organizing networks incorporate speed and robustness to noise along with lenient requirement for a number of clusters specification and training as strengths to deal with characteristics of gene expression data [76].

### 4.8 Soft Clustering

In contrast with hard clustering, soft approaches can assign one gene to more than one cluster (fuzzy assignment), if their expression patterns are similar, allowing genes to be captured in multiple

transcriptional programs and biological processes. Thus, whereas crisp membership is boolean, fuzzy membership is given by a membership weight from 0 to 1. Fuzzy *c*-means (FCM) algorithm was developed by Dunn [36], and enhanced by Bezdek [19] as iterative optimization of an objective function to minimize the variation of objects within clusters [58, 80]. On the other hand, it also has a priori requirement of $c$ value and outliers can be assigned with similar membership in each cluster. Over the years, variants of FCM have been provided, for example, fuzzy *J*-means [13] and FuzzySOM [102]. Whereas the first algorithm is focused on avoiding cluster solution trapped in local minima, the second one is based on SOM for the assignment of cluster centroids into a grid.

Fuzzy clustering by Local Approximation of MEmbership (FLAME) [45] deals with neighborhood relationships and fuzzy membership assignment by local approximation [72]. FLAME can capture non-linear relationships and non-globular clusters, automate definition of the number of clusters, and identify cluster outliers [99]. For more details, e.g., *see* [83].

### 4.9 Multi-Objective Clustering

Multi-objective clustering combines several clustering criteria which are optimized simultaneously. Thus, Handl and Knowles [55] proposed an MOO (Multi-Objective Optimization) clustering algorithm with automatic *k*-determination (MOCK) based on first maximizing cluster compactness, and then maximizing cluster connectivity. MOCK algorithm can automatically find the appropriate partitioning having either hyperspherical-shaped cluster or well-separated clusters [84], and ends up with a number of Pareto optimal solutions. In MOCLE (Multi-Objective CLustering Ensemble) algorithm provided by Fareli et al. [41], aspects from multi-objective methods and cluster ensemble techniques are integrated. MOCLE is applied to the clustering analysis of microarray data with cancer in [119].

Introduced by Bandyopadhyay et al. [10], Archived Multi-Objective Simulated Annealing (AMOSA) algorithm is based on an MOO technique which enhances the performance for problems with many objectives. More recently, $\varepsilon$-AMOSA includes the concept of $\varepsilon$-dominance improving on AMOSA approach [11]. Readers can find an updated review about multi-objective clustering in [103].

## 5 Multiclustering Techniques

In microarray data, the number of genes may range from $10^3$ to $10^4$ (being expected to reach $10^6$), whereas the amount of experimental conditions is usually less than 100 [69]. This characteristic of the microarray data has yielded to apply clustering to both genes and samples by using two-way clustering methods, and [47] analyzed

different aspects of two-way clustering methods, i.e., clustering applied to both genes and samples, under several specifications of the clustering algorithm and similarity measures.

As extension of clustering, these techniques are focused on extracting the genes which are correlated only in a subset of samples, in other words biclustering allows simultaneous clustering of both rows and columns in the data matrix [8]. Considering the kind of the searched biclusters and the mathematical formulation employed to identify them, biclustering techniques are classified into four categories [98].

*Correlation Maximization Biclustering methods* (CMB) seek for subsets of genes and samples where the expression values of the genes (samples) are highly correlated with the samples (genes) [8, 98]. Examples of this class are the CC algorithm [24], and the FLOC algorithm proposed by Yang et al. [133] which overcomes the drawback of random interference caused by masking discovered biclusters in the CC algorithm.

*Variance Minimization Biclustering methods* (VMB) seek for subsets characterized by expression values with low variance throughout the selected genes, conditions, or the whole submatrix [98]. For instance, the conserved gene expression motif (XMOTIF) method proposed by Murali and Kasif [96] aims to find biclusters with constant value at rows [39].

*Two-Way Clustering methods* (TWC) identify biclusters by iteratively carrying out one-way clustering on the genes and samples [98]. As an example of this class, it can be considered the Coupled Two-Way Clustering method (CTWC) proposed by Getz et al. [47] which alternates row and column clustering approaches by using any clustering algorithm.

Finally, *Probabilistic and Generative methods* (PGM) use probabilistic procedures in order to identify genes (samples) similarly expressed across a subset of samples (genes). For example, the cMonkey developed by Reiss et al. [105] models the biclusters by using a Markov chain process.

In regard to triclustering methods, Jiang et al. [70] introduce a set enumeration-based method to mine triclusters from three-dimensional datasets using Pearson correlation coefficient, providing two variants of the method to extract triclusters that are spread over all the time points. TRICLUSTER [136] is another well-known triclustering method that extracts maximal triclusters from three-dimensional datasets using a graph-theoretic approach. gTRICLUSTER [71] uses Spearman rank correlation coefficient to measure local similarities among objects across time points in order to detect more cluster patterns and be more robust to noise. ICSM [5] is a triclustering method that operates on possible pairs of gene-sample planes and detects some initial modules which are further extended to triclusters by using a planar similarity measure.

## 6 Clustering Validation

In unsupervised methods, the assumption of the existence of underlying patterns in the data is an important weakness, and therefore, the resulting clustering should be validated [20]. Apart from determining the number of clusters, assessing the goodness of such a number and the induced cluster assignments is particularly crucial in cluster analysis of gene expression data. As mentioned in [95], the clustering results are especially sensitive to noise and susceptible to overfitting because of the nature of high dimensionality and relatively small sample size in such datasets.

Available validity indexes are used to assess the goodness of clustering obtained, such as Silhouette width [107], Calinski-Harabasz [22], Dunn [37], and Davies-Boudin [30] measurements. These well-known indexes take into consideration the compactness of the objects into the same cluster and the separability between clusters, which are two internal properties for the validation. A brief description of these cluster validity indexes can be found in [85], and from each of them a general rule for interpretation can be derived. The smallest (highest) the value of the last (three first) validity index (indexes), the best the resulting clustering is. It is worth noting the Silhouette width index since it enables to measure the goodness of the clustering outputs for both samples and features, which is given by $sil(x_i) = (b_i - a_i)/\max(a_i, b_i)$, for $i = 1, \ldots, n$, where $a_i$ is the mean distance between the object $x_i$ and the rest in the same cluster, and $b_i$ is the mean distance between the object $x_i$ and the ones of the "nearest neighboring cluster." Its value varies between $-1$ and 1 indicating how well the object $x_i$ has been assigned. A value close to 1 $(-1)$ indicates "well-classified" ("misclassified"). A value close to zero means that the object lies equally far away from the cluster assigned and the nearest neighboring one. Second, the overall goodness of the clustering may be evaluated by the global Silhouette coefficient, which is defined by the mean of the *sil* scores. Kaufman and Rousseeuw [74] suggested the interpretation of the global Silhouette width score as the effectiveness of the clustering structure: there is no substantial clustering structure into $[-1, 0.25]$, it is weak and could be artificial into $(0.25, 0.5]$, there is a reasonable clustering structure into $(0.5, 0.7]$, and a strong clustering structure into $(0.7, 1]$ (*see* Fig. 9).

Furthermore, Lord et al. [85] have recently introduced a new procedure to analyze the robustness of individual objects in a clustering. This procedure provides a novel individual stability index based on a global validity index. Thus, its related cluster and global stability indexes measure the robustness of the clustering and improve the ability to determine the optimal number of clusters. For instance, Lord et al. [85] propose a global stability index

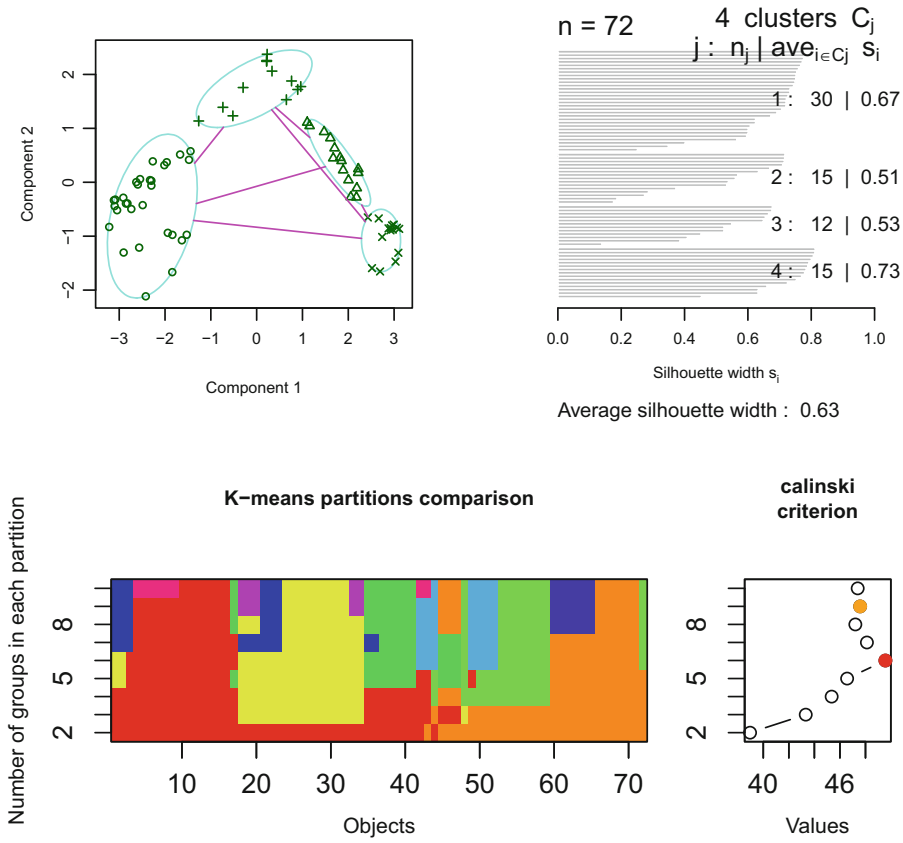


**Fig. 9** Silhouette and Calinski criterion of 3 genes chosen from the gene expression data [50] by selecting complete-linkage clustering

for each one of the above four validity indexes, which allows the comparison among them.

Research in this framework emphasizes the evaluation of the existing clustering quality measures in high-dimensional data, e.g., *see* [123]. More recently, Dash and Misra [29] have conducted a grading approach over five clustering techniques applied to five microarray datasets in order to find a stable technique considering seven validity indexes.

Despite external evaluation being strongly beneficial for assessment of the clustering results, a gold standard is rarely available to carry out it. In this context, indexes for clustering comparison measure the similarity between the clustering and a ground truth partition or gold standard, and the stability of the resulting clusters. For example, the Rand index [104] is given by the ratio of concordant pairs of objects between both of them, i.e., $R = c\binom{n}{2}^{-1}$, where $c$ is the number of pairs of objects that are in the same cluster of the output and in the same cluster of the gold standard plus the number of pairs of objects that are in different clusters of the output and in

different clusters of the gold standard. The Jaccard index [59] measures the stability of each cluster of the output through the proportion of concordant objects between the cluster and the most similar one by a resampling approach. Related indexes involved in such task and other measurements on the accuracy of a discriminant method, e.g., *see* [126] and the references given therein, are usually discussed in classification analysis.

## 7 Summary

The recent explosion of microarray data has posed methodological and computational challenges in cluster analysis due to the inherent nature of such datasets. The potential of these unsupervised methods to detect patterns in microarray data, along with the limitations of the traditional clustering algorithms to handle high-dimensional-low-sample-sized data, has promoted the need to develop sophisticated clustering algorithms that can statistically analyze them.

In this chapter, we have intended to provide an overview on the clustering algorithms of both the first and second generation within the framework of microarray data. Moreover, owing to the many choices of available algorithms, validity indexes used to assess the goodness of clustering solutions are discussed as last stage of the cluster analysis process of microarray data. As reflected in this review, recent techniques developed for performing efficient clusterings on sets with a large number of data have been able to overcome many drawbacks arisen from traditional approaches on microarray data. Currently, developed and adapted clustering algorithms are assisted by available software solutions which spread their use because they make their application easy, fast, and reliable.

## Acknowledgements

This work has been partially supported by Spanish Ministry of Economy and Competitiveness, and the European Regional Development Fund (ERDF) under grants TIN2014-53749-C2-2R and TIN2017-85949-C2-1-R.

### References

1. Achcar F, Camadro JM, Mestivier D (2009) AutoClass@IJM: a powerful tool for Bayesian classification of heterogeneous data in biology. Nucleic Acids Res 37:W63-7. https://doi.org/10.1093/nar/gkp430
2. Aggarwal CC, Reddy CK (2014) Data clustering: algorithms and applications. Chapman and Hall, Boca Raton
3. Aghabozorgi S, Shirkhorshidi AS, Wah T (2015) Time-series clustering - a decade review. Inform Syst 53:16–38

4. Agrawal R, Gehrke J, Gunopulos D *et al* (2005) Automatic subspace clustering of high dimensional data. Data Min Knowl Disc 11:5–33
5. Ahmed HA, Mahanta P, Bhattacharyya DK *et al* (2011) Intersected coexpressed subcube miner: an effective triclustering algorithm. In: Proceedings WICT2011. https://doi.org/10.1109/WICT.2011.6141358
6. Aittokallio T (2010) Dealing with missing values in large-scale studies: microarray data imputation and beyond. Brief Bioinform 11:253–264
7. Alon U, Barkai N, Notterman DA *et al* (1999) Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc Natl Acad Sci USA 96:6745–6750
8. Anand R, Ravichandran S, Chatterjee S (2016) A new method of finding groups of coexpressed genes and conditions of coexpression. BMC Bioinform 17:486. https://doi.org/10.1186/s12859-016-1356-3
9. Ankerst M, Breunig MM, Kriegel H *et al* (1999) OPTICS: ordering points to identify the clustering structure. In: Proceedings ACM SIGMOD 99. https://doi.org/10.1145/304182.304187
10. Bandyopadhyay S, Saha S, Maulik U *et al* (2008) A simulated annealing based multiobjective optimization algorithm: AMOSA. IEEE Trans Evol Comput 12:269–283
11. Bandyopadhyay S, Maulik U, Chakrabortya R (2013) Incorporating $\epsilon$-dominance in AMOSA: application to multiobjective 0/1 knapsack problem and clustering gene expression data. Appl Soft Comput 13:2405–2411
12. Banfield JD, Raftery AE (1993) Model-based Gaussian and non-Gaussian clustering. Biometrics 49:803–822
13. Belacel N, Cuperlović-Culf M, Laflamme M *et al* (2004) Fuzzy J-means and VNS methods for clustering genes from microarray data. Bioinformatics 20:1690–1701
14. Belacel N, Wang Q, Cuperlović-Culf M (2006) Clustering methods for microarray gene expression data. OMICS 10:507–531
15. Bellman R (1961) Adaptive control processes: a guided tour. Princeton University Press, Princeton
16. Ben-Dor A, Shamir R, Yakhini Z (1999) Clustering gene expression patterns. J Comput Biol 6:281–297
17. Berkhin P (2006) A survey of clustering data mining techniques. In: Kogan J, Nicholas C, Teboulle M (eds) Grouping multidimensional data. Springer, Berlin
18. Beyer K, Goldstein J, Ramakrishnan R *et al* (1999) When is nearest neighbor meaningful? In: Beeri C, Buneman P (eds) Proceedings ICDT 99. Springer, Berlin
19. Bezdek JC (1981) Pattern recognition with fuzzy objective function algorithms. Plenum Press, New York
20. Boutros PC, Okey AB (2005) Unsupervised pattern recognition: an introduction to the whys and wherefores of clustering microarray data. Brief Bioinform 6:331–343
21. Brevern AG, Hazout S, Malpertuy A (2004) Influence of microarrays experiments missing values on the stability of gene groups by hierarchical clustering. BMC Bioinform 5:114. https://doi.org/10.1186/1471-2105-5-114
22. Calinski T, Harabasz J (1974) A dendrite method for cluster analysis. Commun Stat 3:1–27
23. Castellanos-Garzón JA, Díaz F (2013) An evolutionary computational model applied to cluster analysis of DNA microarray data. Expert Syst Appl 40:2575–2591
24. Cheng Y, Church GM (2000) Biclustering of expression data. Proc Int Conf Intell Syst Mol Biol 8:93–103
25. Chipman H, Hastie TJ, Tibshirani R (2003) Clustering microarray data. In: Speed T (ed) Statistical analysis of gene expression microarray data. Chapman and Hall, Boca Raton
26. Chipman H, Tibshirani R (2006) Hybrid hierarchical clustering with applications to microarray data. Biostatistics 7:286–301
27. Chiu CC, Chan SY, Wang CC *et al* (2013) Missing value imputation for microarray data: a comprehensive comparison study and a web tool. BMC Syst Biol 7(Suppl 6):S12. https://doi.org/10.1186/1752-0509-7-S6-S12
28. Dahl DB (2006) Model-based clustering for expression data via a Dirichlet process mixture model. In: Do KA, Müller P, Vannucci M (eds) Bayesian inference for gene expression and proteomics. Cambridge University Press, New York
29. Dash R, Misra BB (2018) Performance analysis of clustering techniques over microarray data: a case study. Phys A 493:162–176
30. Davies DL, Bouldin DW (1979) A cluster separation measure. IEEE Trans Pattern Anal Mach Intell 1:224–227
31. Dempster, AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data

via the EM algorithm. J R Stat Soc Ser B 39:1–38
32. D'haeseleer P (2005) How does gene expression clustering work? Nature Biotech 23:1499–1501
33. Do JH, Choi DK (2008) Clustering approaches to identifying gene expression patterns from DNA microarray data. Mol Cells 25:279–288
34. Duda RO, Hart PE, Stork DG (2001) Pattern classification, 2nd edn. Wiley, New York
35. Dudoit S, Fridlyand J, Speed TP (2002) Comparison of discrimination methods for classifications of tumors using gene expression data. J Am Stat Assoc 97:77–87
36. Dunn JC (1973) A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. J Cybern 3:32–57
37. Dunn JC (1974) Well-separated clusters and optimal fuzzy partitions. J Cybern 4:95–104
38. Eisen MB, Spellman PT, Brown PO *et al* (1998) Cluster analysis and display of genome-wide expression patterns. Proc Natl Acad Sci USA 95:14863–14868
39. Eren K, Deveci M, Küçüktunç O *et al* (2013) A comparative analysis of biclustering algorithms for gene expression data. Brief Bioinform 14:279–292
40. Ester M, Kriegel HP, Sander J *et al* (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings KDD 96. AAAI Press, Menlo Park. https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf
41. Faceli K, Carvalho A, Souto M (2007) Multi-objective clustering ensemble. Int J Hybrid Intell Syst 4:145–156
42. Forti A, Foresti GL (2006) Growing hierarchical tree SOM: an unsupervised neural network with dynamic topology. Neural Netw 19:1568–1580
43. Fraley C, Raftery AE (2002) Model-based clustering, discriminant analysis, and density estimation. J Am Stat Assoc 97:611–631
44. Franco M, Vivo JM (2018) Genetic algorithms for parameter estimation in modelling of index returns. Eur J Financ. https://doi.org/10.1080/1351847X.2017.1392332
45. Fu L, Medico E (2007) FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data. BMC Bioinform 8:1. https://doi.org/10.1186/1471-2105-8-3
46. Gentleman R, Ding B, Dudoit S *et al* (2005) Distance measures in DNA microarray data analysis. In: Gentleman R, Carey VJ, Huber W *et al* (eds) Bioinformatics and computational biology solutions using R and Bioconductor. Springer, New York
47. Getz G, Levine E, Domany E (2000) Coupled two-way clustering analysis of gene microarray data. Proc Natl Acad Sci USA 97:12079–12084
48. Gnanadesikan R, Harvey JW, Kettenring JR (1993) Mahalanobis metrics for cluster analysis. Sankhyā A 55:494–505
49. Goil S, Nagesh H, Choudhary A (1999) MAFIA: efficient and scalable subspace clustering for very large data sets. In: Proceedings 5th ACM SIGKDD 99. http://www.academia.edu/download/38278360/goil99mafia.pdf
50. Golub TR, Slonim DK, Tamayo P *et al* (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286:531–537
51. Gollub J, Sherlock G (2006) Clustering microarray data. In: Kimmel AR, Oliver B (eds) DNA microarrays: databases and statistics Part B. Academic Press, San Diego
52. Guha S, Rastogi R, Shim K (2000) ROCK: a robust clustering algorithm for categorical attributes. Inform Syst 25-345–366
53. Guha S, Rastogi R, Shim K (2001) CURE: an efficient clustering algorithm for large databases. Inform Syst 26:35–58
54. Han J, Kamber M, Pei J (2011) Data mining: concepts and techniques, 3rd edn. Morgan Kaufman, San Francisco
55. Handl J, Knowles J (2007) An evolutionary approach to multi-objective clustering. IEEE Trans Evol Comput 11:56–76
56. Hartuv E, Schmitt A, Lange J *et al* (1999) An algorithm for clustering cDNAs for gene expression analysis. In: Proceedings 3rd RECOMB 99. https://doi.org/10.1145/299432.299483
57. Hartuv E, Shamir R (2000) A clustering algorithm based on graph connectivity. Inform Proc Lett 76:175–181
58. Hathaway RJ, Bezdek JC (1985) Local convergence of the fuzzy c-means algorithms. Pattern Recognit 19:477–480
59. Hennig C (2007) Cluster-wise assessment of cluster stability. Comput Stat Data Anal 52:258–271
60. Herrero J, Valencia A, Dopazo J (2001) A hierarchical unsupervised growing neural network for clustering gene expression patterns. Bioinformatics 17(2):126–136
61. Heyer LJ, Kruglyak S, Yooseph S (1999) Exploring expression data: identification and

analysis of coexpressed genes. Genome Res 9:1106–1115

62. Hinneburg A, Keim DA (1998) An efficient approach to clustering in large multimedia databases with noise. In: Proceedings 4th KDD 98, vol 98, pp 58–65
63. Hsu AL, Tang S, Halgamuge SK (2003) An unsupervised hierarchical dynamic self-organizing approach to cancer class discovery and marker gene identification in microarray data. Bioinformatics 19:2131–2140
64. Irigoien I, Mestres F, Arenas C (2013) The depth problem: identifying the most representative units in a data group. IEEE Trans Comput Biol Bioinform 10:161–172
65. Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 31:264–323
66. Jain AK, Dui RPW, Mao J (2000) Statistical pattern recognition: a review. IEEE Trans Pattern Anal Mach Intell 22:4–37
67. Jardine CJ, Jardine N, Sibson R (1967) The structure and construction of taxonomic hierarchies. Math Biosci 1:173–179
68. Jaskowiak PA, Campello RJ, Costa IG (2014) On the selection of appropriate distances for gene expression data clustering. BMC Bioinform 15(S2):S2. https://doi.org/10.1186/1471-2105-15-S2-S2
69. Jiang D, Tang C, Zhang A (2004) Cluster analysis for gene expression data: a survey. IEEE Trans Knowl Data Eng 16:1370–1384
70. Jiang MRCTD, Pei J, Zhang A (2004) Mining coherent gene clusters from gene-sample-time microarray data. In: Proceedings 10th ACM SIGKDD 04. https://doi.org/10.1145/1014052.1014101
71. Jiang H, Zhou S, Guan J *et al* (2006) gTRICLUSTER: a more general and effective 3D clustering algorithm for gene-sampletime microarray data. In: Proceedings BioDM06. Lecture notes in computer science, vol 3916. Springer, Berlin, pp 48–59
72. Kafieh R, Mehridehnavi A (2013) A comprehensive comparison of different clustering methods for reliability analysis of microarray data. J Med Signals Sens 3:22–30
73. Karypis G, Han EH, Kumar V (1999) Chameleon: hierarchical clustering using dynamic modeling. IEEE Comput 32(8):68–75
74. Kaufman L, Rousseeuw PJ (1990) Finding groups in data: an introduction to cluster analysis. Wiley, New York
75. Kennedy J, Eberhart RC (1999) Particle swarm optimization. In: Proceedings 1995 IEEE neural networks. https://doi.org/10.1109/ICNN.1995.488968
76. Kerr G, Ruskin HJ, Crane M *et al* (2008) Techniques for clustering gene expression data. Comput Biol Med 38:283–293
77. Kohonen T (1990) The self-organizing map. Proc IEEE 78(9):1464–1480
78. Korte B, Vygen J (2006) Combinatorial optimization. Theory and algorithms, 3rd edn. Springer, Berlin
79. Krishna K, Murty M (1999) Genetic K-means algorithm. IEEE Trans Syst Man Cybern B 29:433–439
80. Kumar L, Futschik ME (2007) Mfuzz: a software package for soft clustering of microarray data. Bioinformation 2(1):5–7
81. Liew AWC, Law NF, Yan H (2011) Missing value imputation for gene expression data: computational techniques to recover missing data from available information. Brief Bioinform 12:498–513
82. Liu J, Pham TD (2011) Fuzzy clustering for microarray data analysis: a review. Curr Bioinform 6:427–443
83. Liu R, Liu Y, Li Y (2012) An improved method for multi-objective clustering ensemble algorithm. In: Proceedings 2012 IEEE WCCI. https://doi.org/10.1109/CEC.2012.6252972
84. Lord E, Willems M, Lapointe FJ *et al* (2017) Using the stability of objects to determine the number of clusters in datasets. Inform Sci 393:29–46
85. Lu Y, Lu S, Deng Y *et al* (2004) Incremental genetic K-means algorithm and its application in gene expression data analysis. BMC Bioinform 5:172. https://doi.org/10.1186/1471-2105-5-172
86. Lu Y, Lu S, Fotouchi F *et al* (2004) FGKA: a fast genetic K-means clustering algorithm. In: Proceedings 2004 ACM SAC. https://doi.org/10.1145/967900.968029
87. Luo F, Khan L, Bastani F *et al* (2004) A dynamically growing self-organizing tree (DGSOT) for hierarchical clustering gene expression profiles. Bioinformatics 20 (16):2605–2617
88. Macnaughton-Smith P, Williams WT, Dale MB *et al* (1964) Dissimilarity analysis: a new technique of hierarchical sub-division. Nature 202:1034–1035
89. MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings 5th Berkeley Symp Math Stat Prob. https://projecteuclid.org/download/pdf_1/euclid.bsmspp/1200512992
90. Mahalanobis PC (1936) On the generalized distance in statistics. In: Proceedings of

National Institute of Sciences of India. http://www.insa.nic.in/writereaddata/UpLoadedFiles/PINSA/Vol02_1936_1_Art05.pdf

91. McLachlan GJ, Bean RW, Peel D (2002) A mixture model-based approach to the clustering of microarray expression data. Bioinformatics 18(3):413–422
92. McNicholas PD (2016) Model-based clustering. J Classif 33:331–373
93. McNicholas PD, Murphy TB (2010) Model-based clustering of microarray expression data via latent Gaussian mixture models. Bioinformatics 26:2705–2712
94. Monti S, Tamayo P, Mesirov J *et al* (2003) Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. Mach Learn 52:91–118
95. Murali TM, Kasif S (2003) Extracting conserved gene expression motifs from gene expression data. Pac Symp Biocomput 8:77–88
96. Ng RT, Han J (2002) Clarans: a method for clustering objects for spatial data mining. IEEE Trans Knowl Data Eng 14 (5):1003–1016
97. Oghabian A, Kilpinen S, Hautaniemi S *et al* (2014) Biclustering methods: biological relevance and application in gene expression analysis. PLoS One 9:e90801. https://doi.org/10.1371/journal.pone.0090801
98. Oyelade J, Isewon I, Oladipupo F *et al* (2016) Clustering algorithms: their application to gene expression data. Bioinform Biol Insights 10:237–253
99. Pan W, Lin J, Le CT (2002) Model-based cluster analysis of microarray gene-expression data. Genome Biol 3(2):research0009.1-0009.8. http://genomebiology.com/2002/3/2/research/0009.1
100. Parson L, Haque E, Liu H (2004) Subspace clustering for high dimensional data: a review. In: Proceedings 10th ACM SIGKDD. https://doi.org/10.1145/1007730.1007731
101. Pascual-Marqui RD, Pascual-Montano AD, Kochi K *et al* (2001) Smoothly distributed fuzzy *c*-means: a new self-organizing map. Pattern Recognit 34:2395–2402
102. Pizzuti C (2017) Evolutionary computation for community detection in networks: a review. IEEE Trans Evol Comput. https://doi.org/10.1109/TEVC.2017.2737600
103. Rand WM (1971) Objective criteria for the evaluation of clustering methods. J Am Stat Assoc 66:846–850
104. Reiss DJ, Baliga NS, Bonneau R (2006) Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks. BMC Bioinform 7:280–302
105. Röttger R (2016) Clustering of biological datasets in the era of big data. J Integr Bioinform 13:300. https://doi.org/10.2390/biecoll-jib-2016-300
106. Rousseeuw PJ (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math 20:53–65
107. Roy S, Bhattacharyya DK (2007) Data clustering techniques - a review. In: Bhattacharyya DK, Hazarika SM (eds) Networks, security and soft computing: trends and future directions. Narosa Publishing House, New Delhi
108. Saini S, Rani P (2017) A survey on STING and CLIQUE grid based clustering methods. Int J Adv Res Comput Sci 8:1510–1512
109. Saxena S, Purushothaman S, Meghah V *et al* (2016) Role of annexin gene and its regulation during zebrafish caudal fin regeneration. Wound Repair Regen 24:551–559
110. Saxena A, Prasad M, Gupta A *et al* (2017) A review of clustering techniques and developments. Neurocomputing 267:664–681
111. Shannon W, Culverhouse R, Duncan J (2003) Analyzing microarray data using cluster analysis. Pharmacogenomics 4:41–52
112. Sharan R, Shamir R (2000) CLICK: a clustering algorithm with applications to gene expression analysis. Proc Int Conf Intell Syst Mol Biol 8:307–316
113. Sheikholeslami G, Chatterjee S, Zhang A (1998) Wavecluster: a multi-resolution clustering approach for very large spatial databases. In: Proceedings 24th VLDB98. http://www.vldb.org/conf/1998/p428.pdf
114. Sheng Q, Moreau Y, De Smet F *et al* (2005) Advances in cluster analysis of microarray data. In: Azuje F, Dopazo J (eds) Data analysis and visualization in genomics and proteomics. Wiley, West Sussex
115. Shirkhorshidi AS, Aghabozoorgi S, Wah TY (2015) A comparison study on similarity and dissimilarity measures in clustering continuous data. PLoS One 10:e0144059. https://doi.org/10.1371/journal.pone.0144059
116. Sneath PHA, Sokal RR (1973) Numerical taxonomy. The principles and practice of numerical classification. Freeman, San Francisco
117. Steinbach M, Ertöz L, Kumar V (2004) The challenges of clustering high dimensional

data. In: Wille LT (ed) New directions in statistical physics. Springer, Berlin

118. Strehl A, Ghosh J (2002) Cluster ensembles - a knowledge reuse framework for combining multiple partitions. J Mach Learn Res 3:583–618
119. Su M, Chang H (2001) A new model of self-organizing neural networks and its application in data projection. IEEE Trans Neural Netw 12:153–158
120. Tamayo P, Slonim D, Mesirov J *et al* (1999) Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. Proc Natl Acad Sci USA 96:2907–2912
121. Tan PN, Steinbach M, Kumar V (2005) Introduction to data mining. Pearson, Boston
122. Tomasec N, Radovanovic M (2016) Clustering evaluation in high-dimensional data. In: Celebi ME, Aydin K (eds) Unsupervised learning algorithms. Springer, Cham
123. Uma MS, Porkodi R (2016) A survey on clustering algorithm for microarray gene expression data. Int J Recent Innov Trends Comput Commun 4:335–341
124. Van der Lann MJ, Pollard KS, Bryan J (2003) A new partitioning around medoids algorithm. J Stat Comput Simul 73:575–584
125. Vivo JM, Franco M, Vicari D (2018) Rethinking an ROC partial area index for evaluating the classification performance at a high specificity range. Adv Data Anal Classif 12:683–704. https://doi.org/10.107/s11634-017-0295-9
126. Wang W, Yang J, Muntz RR (1997) STING: a statistical information grid approach to spatial data mining. In: Proceedings 23rd VLDB97. http://www.vldb.org/conf/1997/P186.pdf
127. Wang Y, Miller DJ, Clarke R (2008) Approach to working in high-dimensional data spaces: genes expression microarrays. Br J Cancer 98:1023–1028
128. Ward JH (1963) Hierarchical grouping to optimize an objective function. J Am Stat Assoc 58:236–244
129. Wong L (2004) The practical bioinformatician. World Scientific, Singapore
130. Xiao X, Dow ER, Eberhart R *et al* (2003) Gene clustering using self-organizing maps and particle swarm optimization. In: Proceedings 17th IPDPS. https://doi.org/10.1109/IPDPS.2003.1213290
131. Xu D, Tian Y (2015) A comprehensive survey of clustering algorithms. Ann Data Sci 2:165–193
132. Yang J, Wang H, Wang W *et al* (2003) Enhanced biclustering on expression data. In: Proceedings 3rd IEEE BIBE 2003. https://doi.org/10.1109/BIBE.2003.1188969
133. Yu L, Liu H (2003) Feature selection for high-dimensional data: a fast correlation-based filter solution. In: Proceedings 20th ICML-2003. https://www.aaai.org/Papers/ICML/2003/ICML03-111.pdf
134. Zahn CT (1971) Graph-theorical methods for detecting and describing gestalt cluster. IEEE Trans Comput C-20(1):68–86
135. Zhang T, Ramakrishnan R, Livny M (1997) BIRCH: a new data clustering algorithm and its applications. Data Min Knowl Disc 1:141–182
136. Zhao L, Zaki MJ (2005) TriCluster: an effective algorithm for mining coherent clusters in 3D microarray data. In: Proceedings 2005 ACM SIGMOD. https://doi.org/10.1145/1066157.1066236

# Chapter 8

# Classification of Microarray Data

Noelia Sánchez-Maroño, Oscar Fontenla-Romero, and Beatriz Pérez-Sánchez

## Abstract

The automatic classification of DNA microarray data is one of the hot topics in the field of bioinformatics, since it is an effective tool for the diagnosis of diseases in patients. The aim of this chapter is to present the most relevant aspects related to the classification of microarrays. We carried out an analysis of the strategies used for the classification of microarray data and a review of the main methods used in the literature. In addition, other related aspects are addressed as the reduction of dimensionality, to try to eliminate redundant information in genes, or the treatment of imbalanced data and missing of data. To conclude, we present an exhaustive review of the main scientific works in journals to show the most successful techniques applied in this discipline as well as the most used datasets to verify their effectiveness.

**Key words** Classification problems, Classification methods, Microarray, Machine learning, Data preprocessing

## 1 Introduction

Microarray technology is being widely used, among other applications, for the diagnosis of diseases, including cancer. Microarray classification is a supervised learning task that provides the diagnostic category of a tissue sample from its expression array phenotype. For this, it employs labeled data samples and supplies a model that classifies new data samples into different predefined diseases. There are two important challenges that the models used for the classification of microarray data have to face: a low number of instances, often from less than hundreds of patients, and a high dimensionality coming from tens of thousands of genes. These characteristics mean that not all machine learning models are able to obtain a reliable classifier to predict future samples, since many will not adequately generalize from the training data.

In this chapter, we review the machine learning methods most commonly used in the field of microarray classification and, also, related aspects of data preprocessing such as dimensionality

Verónica Bolón-Canedo and Amparo Alonso-Betanzos (eds.), *Microarray Bioinformatics*, Methods in Molecular Biology, vol. 1986, https://doi.org/10.1007/978-1-4939-9442-7_8, 

reduction strategies and the treatment of imbalanced data and missing data. Besides, the most remarkable validation techniques and evaluation measures employed to check the performance of the classifiers are analyzed. Finally, we present an exhaustive analysis of the main recent contributions in the field during the last years and the reference datasets employed in the experimental studies conducted in the literature.

## 2 Classification Problems

The goal in classification is to take an input vector $\mathbf{x}$ and to assign it to one of the $c$ discrete classes. In the most common scenario, the classes are taken to be disjoint, so that each input is assigned to one and only one class. The simplest classification problem consists in distinguishing between two classes ($c = 2$) and it is called binary classification, a more complex task considers a higher number of classes ($c > 2$) and it is known as multiple classes classification. Research has demonstrated convincingly that accurate cancer diagnosis can be achieved by performing microarray data classification, i.e., by constructing classifiers to compare the gene expression profile of a tissue of unknown cancer status to a database stored expression profiles from tissues of known cancer status [1]. Binary classification has attracted most of the interest in the microarray field because many problems try to distinguish between healthy and non-healthy patients. However, trying to classify the different types of cancer is not an easy-to-solve question and different strategies can be adopted. The next subsection copes with these strategies.

### 2.1 Multiple Classes

There are two basic approaches to deal with classifying multiple classes: one is to use classification algorithms that can deal directly with multiple classes, and the alternative is to divide the original problem into several classification problems [2].

Some authors conclude that there is not a method that outperforms every other one, and that the method to be used in order to obtain the best results will depend on the problem, and also on the user-defined constraints, such as the desired level of accuracy, the time available for obtaining a solution, etc. [3]. Classifiers that may cope directly with multiple classes are: extreme learning machine (ELM) [4], ridge regression (RR), and kernel ridge regression (KRR) [5], decision trees and random forest [5], naive Bayes [6], and so on.

There are different schemes to transform the multiple classes problem into several binary problems, known as binarization strategies, the most popular are:

- *One-versus-rest (OVR)*: it transforms a problem with $c$ classes in $c$ binary problems. The elected class represents the positive

samples and the remaining classes the negative ones. It is also known as one-versus-all (OVA).

- *One-versus-one (OVO)*: it generates a classifier to confront each pair of classes, therefore, there are $l = c(c-1)/2$ classifiers.
- The error-correcting output-codes (ECOC) method [7]. In this method the output is transformed by using a matrix $M_{l\times c}$ which columns are the number of classes ($c$) and its rows the number of classifiers ($l$). The authors [7] suggest that the number of classifiers $l$ must be higher than the minimum number necessary to differentiate each class uniquely. The additional bits introduce a redundancy in the classes codification and provide the system the capability to recover from classification errors committed by some of its predictors.

Transforming the multiple class problem into several binary problems is a coding technique that requires a counterpart, i.e., a decoding technique. This decoding technique should integrate the information provided by all the classifiers into a unique solution. Among them, distance-based decoding is the most commonly used scheme [8] when ECOC is applied, mainly using Hamming or Euclidean distance. Besides these distance-based decoding strategies, researchers also proposed some other schemes based on loss function [8]. Majority voting is a common strategy when leading with OVO decomposition, whereas OVR binarization usually requires the base classifiers to produce a real-valued confidence score for achieving a solution, otherwise multiple classes can be predicted for a single sample.

There are many other techniques for generating the binary problems and to obtain the final results, the interested reader may consult [9].

## 3 Overview of Classifiers

Microarray data have two particular characteristics that differentiate it from most of the datasets used in automatic classification tasks: high dimensionality and a small number of instances. This causes that many of the machine learning methods have difficulties to carry out the task of classification in a way that they generalize adequately, mainly due to problems of overfitting the data or the curse of dimensionality. For this reason, many of the methods used in the literature are based on simple models, of linear type, or on nonlinear models designed specifically to alleviate the problems of overfitting. The most used methods in the literature for microarray classification are described in the following subsections.

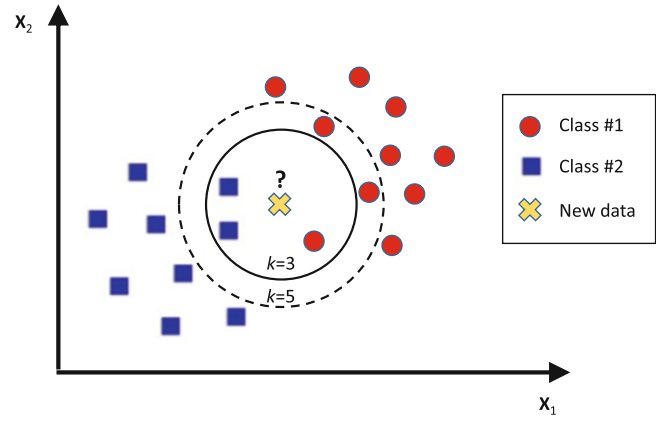


**Fig. 1** Example of classification for *k*-nn algorithm using $k = 3$ and $k = 5$

### 3.1 k-Nearest Neighbors (k-nn)

This algorithm is based on a very simple and intuitive idea, which together with its easy implementation makes it a very extended classification paradigm. The basic idea on which it is based is that a new instance is going to be classified in the most frequent class to which its nearest $k$ neighbors belong. To do this, a metric will be used that allows to measure the distance between data points. This is a lazy algorithm since it does not perform a training process, like the vast majority of learning models, but simply stores in memory the data and its class, provided for the learning of the model, and when it is necessary to classify future samples it calculates the distances of that new data with all those stored in memory. Once the distances have been calculated, they are ordered from least to greatest. The class for the new data is then assigned using the majority class among the $k$ samples with the least distance. Figure 1 graphically shows an example of the algorithm for $k = 3$ and $k = 5$.

An important issue in this method is the determination of the optimal value of $k$. It is empirically ascertained that the percentage of well-classified cases is non-monotonous with respect to $k$.

### 3.2 Support Vector Machines (SVM)

While most of the learning methods focus on minimizing the errors made by the model generated from the training examples (empirical error), the inductive bias associated with the SVMs lies in minimizing the so-called structural risk. The idea is to find a separation hyperplane that is equidistant from the closest examples of each class, in order to achieve what is called a maximum margin on each side of the hyperplane. When defining the hyperplane, only the training examples of each class that fall just at the border of those margins are relevant. These examples are called support vectors. Figure 2 shows an example of two possible hyperplanes (a line in this 2D case) that separate a toy dataset. The line on the right has a greater margin than the line on the left. From a practical point of view, the maximum margin hyperplane has shown to have a good

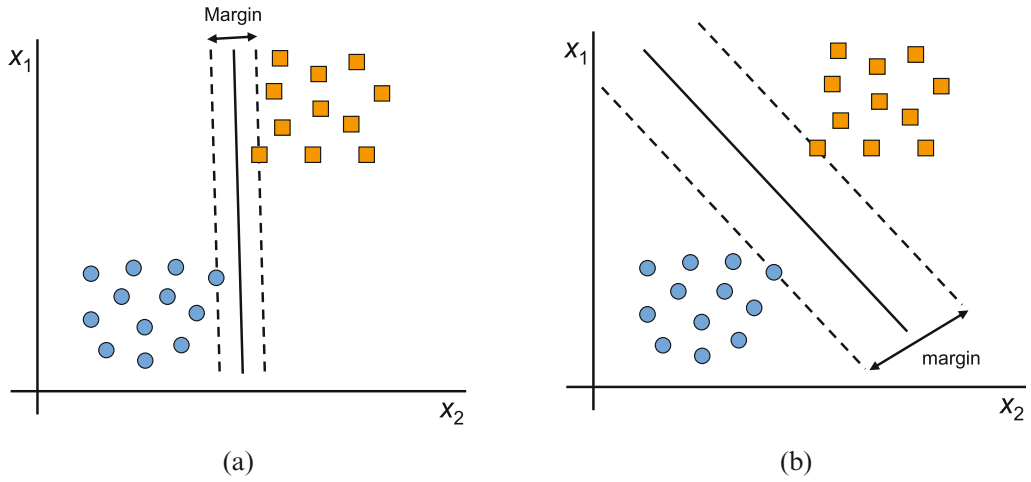


**Fig. 2** Example of two separating lines. The line on the right has a greater margin and then a better generalization capacity. (**a**) A line separating the data with a small margin. (**b**) A line separating the data with a large margin

generalization capacity, avoiding to a large extent the problem of overfitting to the training examples. This is a very desirable characteristic, especially in microarray sets, since they are usually sets composed of few samples and many variables per sample.

In the SVM, the searching process to find the parameters that define the maximum margin hyperplane can be done independently of the dimensionality of the problem to be solved, which is also an advantage in the case of microarrays datasets that usually have a high dimensionality. On one hand, if the dimensionality is low, it is enough to directly solve the associated primal optimization problem. On the other hand, if the dimensionality is very high, it is enough to transform the primal problem into its equivalent dual problem and solve the latter problem.

For high-dimensional problems, as the microarray datasets, sometimes the data are linearly separable. However, in the general case they are not and, then, there is no straight hyperplane that can separate the classes. In the case of problems that are not linearly separable, there is an alternative formulation based on a soft margin that has a hyperparameter, usually called $C$, which measures the trade-off between the training errors and the maximization of the margin. In this case, this formulation allows the decision margin to make a few mistakes, i.e., points inside or on the wrong side of the margin. Then, the hyperparameter $C$ is a regularization term, which provides a way to control overfitting.

In addition, SVMs can be extended to nonlinear models for the classification of nonlinearly separable data. For this, the strategy consists in mapping the original input space into a higher-dimensional feature space in which the training dataset is linearly separable. The process is based on two steps. First, the input data

are transformed into high-dimensional feature vectors and, later, SVM is used to find the optimal hyperplane in the feature space. Thus, although the obtained hyperplane is a linear separating function in the new feature space, it is a nonlinear function in the original input space. This transformation is achieved using what is known as the "the kernel trick." The choice of a proper kernel function is important, since it defines the feature space in which the training instances should be classified.

For all the aforementioned, the SVM is the most popular classification method for microarray datasets as it obtains good generalization results, mainly because it avoids overfitting by choosing the maximum margin hyperplane among all those that can separate the data in the feature space. In addition, the SVM uses a straightforward learning algorithm that solves a convex optimization problem that contains a unique minimum avoiding the problem of local minima. Therefore, for a given dataset, the training process always converges to the same solution regardless of the starting conditions used, unlike other learning methods as, for example, artificial neural networks.

### 3.3 Decision Trees and Random Forests

Decision trees are one of the most popular classification models in bioinformatics, especially due to their capacity to represent the results in a format which is easy to interpret for humans. The aim of this method is to generate a tree whose leaves are linked with a specific value of the class and whose inner nodes represent descriptive attributes. Given an inner node, each child corresponds to an alternative value of the associated attribute. To build decision trees, such as C4.5 [10] or CART [11], usually a top-down heuristic search with recursive partitioning is used. The most important step in tree growing is to select the best attribute to split a node. Starting from a heterogeneous training set, in terms of the variation in the class label, each attribute is independently evaluated using a statistic to determine how well it classifies the training samples. With this purpose, several measures have been designed to evaluate the degree of inhomogeneity, or impurity, in a set. For decision trees, the two most common measures are entropy and the Gini index. The best attribute is selected using any of these measures to split the training samples. This process is recursively repeated to split the descendant nodes until some pre-established stopping criteria is achieved. This search algorithm is greedy since it never backtracks to reconsider its previous decisions. Usually this phase is followed by a bottom-up pruning step to avoid overfitting.

Although decision trees generate efficient models, they are unstable and tend to overfit training data, which can give poor generalization results. Because of this, they are often used in classifier ensembles. Random forests [12] are a type of nonparametric predictive models consisting of an ensemble of classification trees, where each tree has been trained using a bootstrap sample of

individuals from the data. They are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. Random forests classify an instance by assigning it to each tree of the forest and each tree gives an individual classification. Subsequently, the forest chooses the class with the most votes from all the trees. Unlike classical decision trees, in random forest there is no need to prune trees since the ensemble and bootstrapping schemes help them to overcome overfitting issues.

Each tree in the forest is constructed, from data having $n$ instances and $m$ variables, as follows:

1. From the whole dataset, select a training set by choosing $n$ instances with replacement.
2. At each node in the tree, randomly select $s$ variables from the $m$ variables in the dataset, where $s << m$. The value of $s$ remains constant throughout the entire process.
3. Get the best split at each node from among the subset of $s$ variables selected in the previous step.
4. Iterate steps 2 and 3 until the tree is fully grown.

The random sampling and ensemble strategies utilized by random forest allow it to get accurate predictions as well as achieving a good generalization capacity. Also, unlike many of the machine learning models, it allows the use of various types of variables (continuous, categorical, etc.) and its results are interpretable. This is an important characteristic, since in many of the bioinformatics problems the interpretability of predictive algorithms is considered as important as the prediction accuracy.

## 4 Data Preprocessing

In machine learning, there are many preprocessing techniques that can be applied to improve the performance results of any classifier. In this section we focus on those employed for microarray datasets.

### 4.1 Dimensionality Reduction

The high dimensionality of data has an important impact in learning algorithms, since they degrade their performance when a number of irrelevant and redundant features are present. In fact, this phenomenon is known as the *curse of dimensionality*, because unnecessary features increase the size of the search space and make generalization more difficult.

Dimensionality reduction is the transformation of high-dimensional data into a meaningful representation of reduced dimensionality. Ideally, the reduced representation should have a dimensionality that corresponds to the intrinsic dimensionality of the data, i.e., the minimum number of parameters needed to account for the observed properties of the data. This dimensionality

reduction helps to mitigate the curse of dimensionality and other undesired properties of high-dimensional spaces [13]. Another important reason to reduce dimensionality is to help biologists to identify the underlying mechanism that relates gene expression to diseases. Dimensionality reduction techniques can be divided into feature extraction and feature selection. Feature extraction (FE) is a process that extracts a set of new features from the original features through some functional mapping [14]. Classical examples are principal component analysis (PCA) and multidimensional scaling (MDS) [15].

Feature selection (FS) is defined as the process of identifying and removing irrelevant and redundant features from the training data, so that the learning algorithm focuses only on those aspects of the training data useful for analysis and future prediction. This reduction in the input dimensionality implies, most of the time, an improvement in the performance. There are three main models that deal with feature selection: filter methods, wrapper methods, and embedded methods. While wrapper models involve optimizing a predictor as part of the selection process, filter models rely on the general characteristics of the training data to select features with independence of any predictor. The embedded methods generally use machine learning models for classification, and then an optimal subset of features is built by the classifier algorithm. Although wrapper model tends to obtain better performances, it is very time consuming and has the risk of overfitting due to the reduced number of instances of microarray data and the small ratio between number of samples and number of features [16].

In the past several decades, many dimensionality reduction techniques have been proposed. Focus on feature extraction, the paper by Van [13], presents a comparative study of the most important linear dimensionality reduction technique (PCA) and 12 frontranked nonlinear dimensionality reduction techniques. After an experimental study using artificial and real data, they concluded that nonlinear techniques for dimensionality reduction are, despite their large variance, often not capable of outperforming traditional linear techniques such as PCA [13].

New feature selection methods are constantly being developed so there is a wide suite available to researchers. Traditionally, the most employed gene selection methods fall into the filter approach. However, it is also worth noticing that the review of up-to-date literature has shown a tendency to mix algorithms, in the form of either hybrid methods or ensemble methods. To know the most recent advances in the feature selection field, the interested reader can consult the paper by Bolón-Canedo et al. [17].

### 4.2 Class Representation

In an ideal situation, all classes of a problem would be equally represented, i.e., each class would have the same number of samples. However, in real situations this frequently does not happen.

Microarray data are known to have skewed class distribution which is called class imbalance problem. For instance, lung dataset—formed by 181 instances and 12,533 features—has 83% samples in one class and just 17% in the other one. This dataset is formed by joining the original datasets: (1) Lung-test (149 instances) and (2) Lung-train (32 instances). Lung-train set is completely balanced (50–50); however, the lung-test is extremely skewed (90–10). Therefore, a classifier with good accuracy level in the training phase may lead to very poor performance results in the test set. The problem becomes more complex when dealing with multiple classes, where again healthy patients outnumber cancer patients or a specific type of cancer is much more uncommon than the others. When the number of classes is very large, there are many probabilities that one (or several) of them have a very low representativeness and its correct classification becomes a challenge. The class imbalance problem has been recognized as a crucial problem in machine learning because such a problem is encountered in a large number of domains and, in certain cases, it causes seriously negative effects on the performance of classifiers. To deal with this issue different strategies can be considered [18], presenting all is beyond the scope of this chapter, we will focus on those most applied to microarray datasets:

- Random oversampling and undersampling: Oversampling consists in augmenting the original set *S* by replicating a set of selected samples from the minority class and adding them to *S*. In this way, the number of total examples in *S* is increased and the class distribution balance of *S* is adjusted accordingly. Whereas oversampling appends data to the original dataset, random undersampling removes data from the original dataset. Analogously, we randomly select a set of majority class examples and remove these samples from *S*. Both methods have their drawbacks. In the case of undersampling, removing examples from the majority class may cause the classifier to miss important concepts pertaining to the majority class. In the case of oversampling, multiple instances of certain examples become *tied* leading to overfitting.
- Synthetic minority over-sampling technique (SMOTE): The SMOTE algorithm creates artificial data based on the feature space similarities between existing minority examples. Specifically, the minority class is oversampled by taking each minority class sample and introducing *synthetic* examples along the line segments joining any/all of the $k$ minority class nearest neighbors. Depending upon the amount of oversampling required, neighbors from the $k$ nearest neighbors are randomly chosen [19].

### 4.3 Missing Data

Although DNA microarray is a mature technology, expression data can contain missing values due to several reasons as, for example, scratches or spots on the slide or fabrication errors. To solve this problem, several solutions can be proposed. The most obvious solution would be to repeat the microarray experiments, but it should be avoided because it is the most expensive option. Another rather simple solution would be to remove genes containing missing values, but in that case a lot of relevant information could be lost. The most beneficial solution, but at the same time the most sophisticated, is the one that uses imputation procedures to estimate missing values.

In the literature, several methods have been proposed to be able to impute missing values in a general way and some others for microarray data in particular. The imputation of missing data supposes the use of information on the rest of the data to be able to estimate the deficiencies. For this, there are two types of information that is usually used. The first type of information consists of the correlation structure between the entries in the data matrix. The correlation between instances exists due to the fact that genes involved in similar cellular processes usually have similar expression profiles. In this sense, some approaches estimate the missing values based on the overall correlation information obtained from the complete dataset, while others use only a local structure of the data. Likewise, another type of correlation can exist between variables (genes) since it is expected that the set of genes behaves similarly in similar situations. The second type of information is the knowledge that is available about the domain or the biological processes that generate the data. This knowledge can be used to obtain more plausible estimates. In this sense, several of the most current data imputation algorithms have tried to incorporate this information on the underlying processes to try to improve the imputation performance [20, 21]. Recent surveys studies with a comprehensive review and main techniques for microarray missing value imputation can be found in the work by Chiu et al. [22] and [23].

## 5 Evaluation Procedures

The number of papers using microarray datasets has not stopped growing in recent years. Mainly for two reasons, on the one hand, there are more and more microarray datasets on specialized platforms (ArrayExpress[1] or Gene Expression Omnibus[2]). On the other hand, the intrinsic characteristics of these datasets pose a

[1] http://www.ebi.ac.uk/arrayexpress/.
[2] http://www.ncbi.nlm.nih.gov/geo/.

**Table 1**
**Confusion matrix**

| | Actual negative | Actual positive |
|---|---|---|
| Predicted negative | True negative (TN) | False negative (FN) |
| Predicted positive | False positive (FP) | True positive (TP) |

challenge for learning methods where new methods are emerging trying to solve them. Therefore, whenever a novel method is presented or a new dataset is created, it is important to test it properly. In this section we present the evaluation measures and validation techniques for these datasets.

### 5.1 Evaluation Measures

In pattern recognition, information retrieval, and binary classification, many evaluation measures are defined from a confusion matrix or contingency table. The structure of this matrix can be seen in Table 1.

Recent papers have followed the convention that confusion matrix rows correspond to alternative predicted classes, while columns correspond to actual classes. Using this table, the following measures are defined:

$$\text{Accuracy} = (\text{TN} + \text{TP})/(\text{TN} + \text{TP} + \text{FN} + \text{FP})$$
$$\text{Precision or Positive Predictive Value (PPV)} = \text{TP}/(\text{TP} + \text{FP})$$
$$\text{Sensitivity or True Positive Rate (TPR)} = \text{TP}/(\text{TP} + \text{FN})$$
$$\text{Specificity (SPC) or True Negative Rate} = \text{TN}/(\text{TN} + \text{FP})$$
$$\text{Fall-out or False Positive Rate (FPR)} = \text{FP}/(\text{FP} + \text{TN}) = 1 - \text{SPC}$$

As commented in Subheading 4.2, some microarray datasets are highly unbalanced, so there exists a class with most of samples. In these situations, the design of a high accuracy classifier is straightforward: just classifying every new case as belonging to the majority class. However, accurate classification of minority class cases (usually non-healthy patients) is frequently the major objective in microarray datasets. Then, in order to evaluate the performance of any classifier applied on these datasets it is not valid to compute only accuracy, at least the specificity—focused on the minority class—should also be calculated, although most common is to include these three measures: accuracy, sensitivity, and specificity. Another important measure of the performance is the area under the ROC curve (AUC). The receiver operating characteristic (ROC) curve is created by plotting the sensitivity against the false positive rate (FPR) at various threshold settings. In binary classification, the class prediction for each instance is often made based on a *score (x)* computed for that instance. Given a threshold parameter $T$, the instance is assigned to one class if $x > T$, and otherwise to the other. Varying this threshold $T$,

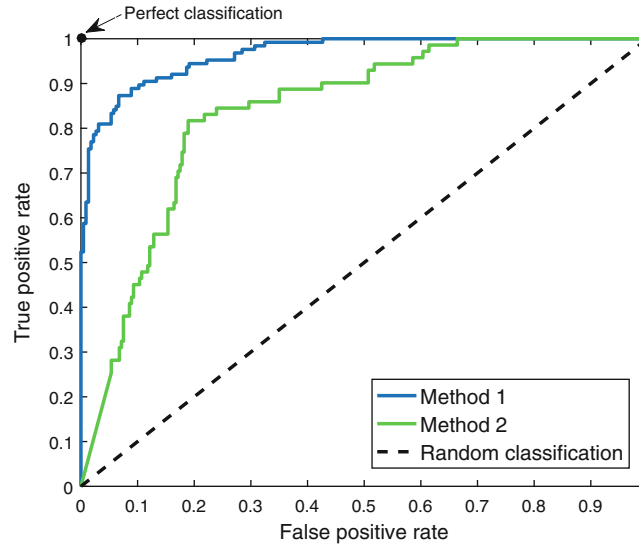


**Fig. 3** ROC curve of two different methods

different values of sensitivity and specificity are achieved and this variation is reflected in a ROC curve. Figure 3 illustrates an example of this curve by showing the hypothetical results of two methods (labeled Method 1 and 2). The best possible prediction method would yield a point in the upper left corner of the ROC space, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives). A random guess would give a point along the dotted diagonal line. The performance of method 1 is better than the method 2, because its curve gets closer to the perfect score. Moreover, the area under the curve (AUC) for method 1, i.e., the surface between the curve and the $X$ axis, is larger than the AUC for method 2.

The previous example using the ROC curve (*see* Fig. 3) provides a way for comparing two learning algorithms (or more) on a single dataset. However, usually we want to analyze the performance of a new classifier (or a part of it or some new pre- or postprocessing step) comparing the results achieved against other methods over different datasets. Then, to determine which classifier is the best, common strategies are to compute the average accuracy or the average position in a ranking (after obtaining a performance ordered list of classifiers for each dataset). However, statistical test is highly recommended and Demšar [24] suggested a set of simple, yet safe and robust nonparametric tests for statistical comparisons of classifiers: the Wilcoxon signed ranks test for comparison of two classifiers and the Friedman test with the corresponding post-hoc tests for comparison of more classifiers over multiple datasets.

### 5.2 Validation Techniques

To evaluate the goodness of a model, it is necessary to have an independent test set with data which have not seen by the classification task. In some cases, the data come originally distributed into train and test sets, so the training set is usually employed to perform the learning model and the test set is used to evaluate its appropriateness. However, not all the datasets found in the literature are originally partitioned. For overcoming this issue, there exist several validation techniques, where the most used ones in the microarray domain are:

- $k$-fold cross-validation ($k$-fold CV). The data ($D$) is partitioned into $k$ non-overlapping subsets $D_1, D_2, \ldots, D_k$ of roughly equal size. The learner is trained on $k - 1$ of these subsets combined together and then applied to the remaining subset to obtain an estimate of the prediction error. This process is repeated in turn for each of the $k$ subsets, and the cross-validation error is given by the average of these $k$ estimates [25].
- Leave-one-out cross-validation (LOOCV): it is a variant of $k$-fold CV where $k$ is the number of samples, so a single observation is left out each time [25].
- Bootstrap. It is a general resampling technique. A *bootstrap* sample consists of $n$ (being $n$ the number of samples) equally likely draws with replacement from the original data. Therefore, some of the samples will appear multiple times, whereas others will not appear at all. The learner is designed on the bootstrap sample and tested on the left-out data points. The error is approximated by a sample mean based on independent replicates (usually between 25 and 100). There exist some famous variants of the method such as *balanced bootstrap* or *0.632 bootstrap* [26].
- Holdout validation (HO). This technique consists of randomly splitting the data into a disjoint pair train-test. A common partition is to use $2/3$ for training and $1/3$ for testing.

A key point for microarray classification is that error estimation is greatly impacted by small samples [27]. The behavior of cross-validation for very small samples has been thoroughly studied in Braga-Neto and Dougherty [28] who did not even find substantial differences, in terms of decreased variance, among the cross-validated variants (leave-one-out, 5- and 10-fold, stratified and repeated cross-validation). In the review by Bolon-Canedo et al. [27], they test the performance of regular cross-validation against DOB-SCV (distribution optimally balanced stratified cross-validation) concluding that, on average, the latter obtains better results than the former. Bootstrap estimation procedures are smoothed versions of cross-validation to reduce the variability of performance estimates. They come at the price of a high computational cost and an increased bias [28]. It seems that there is not a consensus in determining a validation technique for this field. In

fact, reviewing the recent literature one can find examples of the four methods described above. k-fold cross-validation is a common choice [29–46]. There are also many representatives of leave-one-out cross-validation [38, 47–57], as well as holdout validation [42, 58–66]. Although in small number, there are also some research papers using the original division of the datasets [5, 63, 67–69]. Finally, bootstrap sampling seems less used [70, 71], probably due to its high computational cost.

## 6 Recent Contributions

At present there are numerous classification methods that have been applied in the field of microarrays, in this section we have tried to summarize them including the preprocessing and validation techniques used. Table 2 illustrates the most common machine learning methods found in the literature from 2010, where SVM stands out as the most used method. For the sake of brevity Table 2 cannot reflect all applied machine learning methods. However, for those researchers who want to test some new method in this field, next we show some of the less common applied classifiers (it is not the intention of the authors to create an exhaustive list since the literature in this field is extremely extensive): logistic regression (LR), evolutionary generalized radial basis function (EGRBF) [58], multi-logistic (MultiLog), simple logistic (SimpleLog), logistic model tree (LMT) [58, 72], ridge logistic regression (RLR) [5, 70], weighted voting (WV) [49], LDA with genetic algorithm (LDA + GA) [30], intelligent dynamic genetic algorithm (IDGA) combined with different classifiers (SVM, KNN, and NB) [47], lattice neural network with dendritic processing (LNNDP), Bayes net [63], expectation propagation (EP), Bayesian model with EP, Markov chain Monte Carlo (MCMC), diagonal linear discriminant analysis (DLDA), nearest shrunken centroids (NSC), relevance vector machine (RVM) [59], linear discriminant (LD), repeated incremental pruning to produce error reduction (RIPPER) [64], combined penalization least squares (CP-LS), elastic net (Enet), sparse logistic regression (SLR) [57], adaptive huberized support vector machine (AHSVM), Golub's method [61], case-based reasoning method (MicroCBR) [62], wavelet neural network (WNN) [46], binary particle swarm optimization (BPSO), particle swarm optimization (PSO) based DT, Taguchi chaotic binary particle swarm optimization (CFS-TCBPSO-1NN), Markov blanket-embedded genetic algorithm (MBEGA) [36], IB1 [29], Ibk [43], probabilistic neural network (PNN) [54], multitest decision tree (MTDT) [33], Laplace naive Bayes model with mean shrinkage (LNB-MS), Gaussian naive Bayes model with mean shrinkage (GNBMS), distribution-based classification with feature selection based on t-statistic (DBC-T), margin influence analysis (MIA),

**Table 2**
**Summary of the most used techniques from 2010 for the classification of microarrays**

| Classifier | Task | Preprocessing | Validation | References |
|---|---|---|---|---|
| SVM | Binary, multi, one-class | Dissimilarity space, double-bounded tree-connected Isomap (dbt-Isomap), FE, FS, FS with artificial bee colony (ABC), FS based on cat swarm optimization (CSO), FS using evolutionary algorithms (EA), fuzzy preference-based rough set (FPRS), kernelized fuzzy rough set (KFRS), microarray gene selection based on ant colony (MGSACO), PCA-log regression, texture descriptors, Wavelet decomposition | Test set, HO, CV, LOOCV, Bootstrap | [5, 8, 29, 32, 36, 41, 43–45, 49, 53, 55, 58–61, 63–66, 68, 69, 71, 75–79] |
| KNN | Binary, multi | FS, KFRS, FPRS, information theory and approximate Markov blanket | HO, CV, LOOCV | [8, 37, 40, 42, 45, 49, 52, 54, 59, 60, 64–66, 75, 78] |
| Random forest | Binary, multi | FS, FS based on CSO, guided regularized random forest (GRRF) | Test set, CV | [5, 31, 34, 36] |
| Decision trees | Binary, multi | FS, GRRF, MGSACO | HO, CV | [16, 29, 33, 34, 42, 43, 58, 64, 72, 79] |
| Extreme learning machine (ELM) | Binary, multi | FS, PCA-log regression | Test set, HO, CV | [4, 39, 63, 80] |
| Naive Bayes (NB) | Binary, multi | FE (ICA), FS, FPRS, KFRS, PCA-log regression | Test set, HO, CV | [29, 40, 43, 45, 63, 76, 81, 82] |
| Multilayer perceptron (MLP) | Binary, multi | FS, FS with ABC, PCA-log regression | Test set, HO, CV, LOOCV | [42, 54, 58, 63, 68, 78] |
| Radial basis function (RBF) | Binary | FS with ABC, PCA-log regression | Test set and HO | [58, 63, 68, 72] |
| Lasso | Binary | None | HO, bootstrap | [59, 70, 73] |

(continued)

**Table 2**
**(continued)**

| Classifier | Task | Preprocessing | Validation | References |
|---|---|---|---|---|
| AdaBoost (AB) | Binary | FS | HO, CV | [42, 44, 58] |
| Linear discriminant analysis (LDA) | Binary | Genetic algorithm (GA), FS | CV, LOOCV, bootstrap | [37, 54, 71] |

**Table 3**
**Summary of the most used datasets for microarrays as well as the classifiers used in each case**

| Datasets | Classifiers | References |
|---|---|---|
| Breast | AB, Bayesian model with EP, BPSO, CFS-TCBPSO-1NN, DLDA, DT, ELM, EP, EGRBF, FVQIT, GA-based SVM, KNN, LASSO, LD, logistic model tree (LMT), LR, MBEGA,MCMC, MLP, MultiLog, NB, NSC, PSO-based DT, RBF, RLR, RIPPER, RF, RFE, RVM, SimpleLog, SRKNN, SVM | [5, 33, 35, 36, 40–42, 51, 52, 56, 58, 59, 64, 68, 74, 80] |
| Colon | AB, Bayesian model with EP, BPSO, CFS-TCBPSO-1NN, DBC-T, DLDA, ensemble fuzzy RF, ELM, EP, FLD, GRBF, GNB, KNN, LASSO, LNB, LNB-T, MBEGA, MCMC, MIA, MLP, MSVM-RFE, MultiLog, NB-MS, NSC, PAC Bayes, PSO-based DT, RBF, RF, RFE, RLR, RVM, sequential random KNN, SimpleLog, SVM | [5, 30, 31, 35, 36, 40, 41, 44, 55, 56, 59, 65, 72, 74, 75, 78, 80] |
| Leukemia | IDGA + SVM, KNN, NB, RLR, KNN, SVM, RF, LNNDP, ELM, Bayes Net, BPSO, PSO-based DT, MBEGA, CFS-TCBPSO-1NN, DT, IB1, GA, LDA, MLP, MapReduce-based proximal SVM (mrPSVM), SRKNN, WV, Bayesian model with EP, EP, MCMC, DLDA, RFE, RVM, NSC, LASSO, RIPPER, FVQIT, Golub's method, AB, RBF, MultiLog, SimpleLog, LMT, RLR, dbt-Isomap, LNB-MS, GNB-MS, LNB-T, DBC-T, MSVM-RFE, MIA, LNB, GNB, GRBF, SLiM | [5, 16, 29, 30, 32, 33, 36–38, 40, 42, 44, 45, 47–49, 54, 55, 58–61, 63–65, 67– 76, 79–83] |

(continued)

**Table 3 (continued)**

| Datasets | Classifiers | References |
|---|---|---|
| Lung | AB, BPSO, CFS-TCBPSO-1NN, DBC-T, DT, EA + SVM, ELM, ESVM, FVQIT, GA-based ensemble SVM, GNB-MS, IB1, KNN, LD, LDA + GA, GNB, Lasso, LMT, LNB, LNB-MS, LNB-T, MBEGA, MC-SVM, MGSACO, MIA, MLP, MSVM-RFE, MultiLog, NB, PSO-based DT, RBF, RF, RIPPER, SimpleLog, SLiM, SVM | [16, 29, 30, 35, 36, 38, 42, 52, 53, 56, 58, 64–66, 72, 73, 75, 77, 78, 80, 81] |
| Lymphoma | Bayes net, Bayesian model with EP, BPSO, CFS-TCBPSO-1NN, DLDA, DT, ELM, ensemble fuzzy RF, EP, GSA, KNN, IB1, LASSO, LNNDP, MBEGA, MCMC, MLP, NB, NSC, PSO-based DT, RBF, RF, RFE, RVM, SVM | [16, 31, 36, 48, 59, 63] |
| Ovarian | AB, Bayesian model with EP, BPSO, CFS-TCBPSO-1NN, DLDA, DT, FVQIT, IB1, KNN, LASSO, LDA + GA, MLP, NB, PSO-based decision trees, MBEGA, MCMC, MLP, NB, NSC, EP, RF, RFE, RVM, SVM | [16, 29, 30, 36, 41, 42, 59, 64, 75, 78] |
| Prostate | AB, DT, EA + SVM, ensemble fuzzy RF, ESVM FLD, forest classification tree, forest SVM, FVQIT, GA-based ensemble SVM, IB1, KNN, LD, LDA + GA, MC-SVM, MGSACO, MLP, NB, RIPPER, SLiM, SRKNN, SVM | [16, 29–31, 40–42, 52, 56, 64, 67, 74, 75, 77– 79, 84] |
| SRBCT | ELM, IDGA + SVM, KNN, NB, KNN, RF, RLR, SVM | [4, 5, 47, 65–67, 80] |

Laplace naive Bayes model (LNB), Gaussian naive Bayes model (GNB) [73], genetic swarm algorithm (GSA) [48], generalized radial basis function neural networks (GRBFNNs) [72], probably approximately correct (PAC) Bayes [44], SLiM (integrates feature selection with classification) [67], and sequential random k-nearest neighbors (SRKNN) [74].

Analogously, although new datasets are constantly appearing, there are some *classical* datasets that have been tested with numerous techniques shown in Table 3. It is worth mentioning that there are many variants of some datasets (adding/removing features and/or samples) that have not been considered when constructing Table 3. The researcher should bear in mind that a slight variation in the number of samples or features can lead to very different performance results, so for any comparative study, it is mandatory to look carefully the characteristics of each dataset.

## References


1. Peng Y (2006) A novel ensemble machine learning for robust microarray data classification. Comput Biol Med 36(6):553–573
2. Sánchez-Maroño N, Alonso-Betanzos A, García-González P, Bolón-Canedo V (2010) Multiclass classifiers vs multiple binary classifiers using filters for feature selection. In: The 2010 international joint conference on neural networks (IJCNN). IEEE, Piscataway, pp 1–8
3. Golestani A, Ali Amiri KA, Jahed Motlagh MR (2007) A novel adaptive-boost-based strategy for combining classifiers using diversity concept. In: 6th IEEE/ACIS international conference on computer and information science, 2007, ICIS 2007. IEEE, Piscataway, pp 128–134
4. Liu Z, Tang D, Cai Y, Wang R, Chen F (2017) A hybrid method based on ensemble WELM for handling multi class imbalance in cancer microarray data. Neurocomputing 266:641–650
5. Mohapatra P, Chakravarty S, Dash PK (2016) Microarray medical data classification using kernel ridge regression and modified cat swarm optimization based gene selection system. Swarm Evol Comput 28:144–160
6. Friedman N, Geiger D, Goldszmidt M (1997) Bayesian network classifiers. Mach Learn 29 (2–3):131–163
7. Dietterich TG, Bakiri G (1995) Solving multiclass learning problems via error-correcting output codes. J Artif Intell Res 2:263–286
8. Liu K-H, Zeng Z-H, Ng VTY (2016) A hierarchical ensemble of ECOC for cancer classification based on multi-class microarray data. Inf Sci 349:102–118
9. Lorena AC, De Carvalho ACPLF, Gama JMP (2008) A review on the combination of binary classifiers in multiclass problems. Artif Intell Rev 30(1–4):19
10. Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann, San Mateo
11. Breiman L, Friedman J, Olshen R, Stone C (1984) Classification and regression trees. Wadsworth International Group, Belmont
12. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
13. Van Der Maaten L, Postma E, Van den Herik J (2009) Dimensionality reduction: a comparative. J Mach Learn Res 10:66–71
14. Liu H, Motoda H (1998) Feature extraction, construction and selection: a data mining perspective, vol 453. Springer Science & Business Media, New York
15. Guyon I, Gunn S, Nikravesh M, Zadeh LA (2008) Feature extraction: foundations and applications, vol 207. Springer, Berlin
16. Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A (2012) An ensemble of filters and classifiers for microarray data classification. Pattern Recogn 45(1):531–539
17. Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A (2015) Recent advances and emerging challenges of feature selection in the context of big data. Knowl-Based Syst 86:33–45
18. He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21 (9):1263–1284
19. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority oversampling technique. J Artif Intell Res 16:321–357
20. Gan X, Liew AW-C, Yan H (2006) Microarray missing data imputation based on a set theoretic framework and biological knowledge. Nucleic Acids Res 34(5):1608–1619
21. Xiang Q, Dai X, Deng Y, He C, Wang J, Feng J, Dai Z (2008) Missing value imputation for microarray gene expression data using histone acetylation information. BMC Bioinformatics 9(1):252
22. Chiu C-C, Chan S-Y, Wang C-C, Wu W-S (2013) Missing value imputation for microarray data: a comprehensive comparison study and a web tool. BMC Syst Biol 7(6):S12
23. Liew AW-C, Law N-F, Yan H (2011) Missing value imputation for gene expression data: computational techniques to recover missing data from available information. Brief Bioinform 12(5):498–513
24. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7(Jan):1–30
25. Bramer M (2007) Principles of data mining, vol 180. Springer, London
26. Efron B, Tibshirani RJ (1994) An introduction to the bootstrap. CRC Press, Boca Raton
27. Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A, Benítez JM, Herrera F (2014) A review of microarray datasets and applied feature selection methods. Inf Sci 282:111–135
28. Braga-Neto UM, Dougherty ER (2004) Is cross-validation valid for small-sample microarray classification? Bioinformatics 20 (3):374–380
29. Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A (2014) Data classification using an ensemble of filters. Neurocomputing 135:13–20

30. Huerta EB, Duval B, Hao J-K (2010) A hybrid LDA and genetic algorithm for gene selection and classification of microarray data. Neurocomputing 73(13):2375–2383
31. Cadenas JM, Garrido MC, Martínez R (2013) Feature subset selection filter-wrapper based on low quality data. Expert Syst Appl 40 (16):6241–6252
32. Cao J, Zhang L, Wang B, Li F, Yang J (2015) A fast gene selection method for multi-cancer classification using multiple support vector data description. J Biomed Inform 53:381–389
33. Czajkowski M, Grześ M, Kretowski M (2014) Multi-test decision tree and its application to microarray data classification. Artif Intell Med 61(1):35–44
34. Deng H, Runger G (2013) Gene selection with guided regularized random forest. Pattern Recogn 46(12):3483–3489
35. Guo S, Guo D, Chen L, Jiang Q (2016) A centroid-based gene selection method for microarray data classification. J Theor Biol 400:32–41
36. Jain I, Jain VK, Jain R (2018) Correlation feature selection based improved-binary particle swarm optimization for gene selection and cancer classification. Appl Soft Comput 62:203–215
37. Karimi S, Farrokhnia M (2014) Leukemia and small round blue-cell tumor cancer detection using microarray gene expression data set: Combining data dimension reduction and variable selection technique. Chemom Intell Lab Syst 139:6–14
38. Pramod Kumar P, Vadakkepat P, Poh LA (2011) Fuzzy-rough discriminative feature selection and classification algorithm, with application to microarray and image datasets. Appl Soft Comput 11(4):3429–3440
39. Lee K, Man Z, Wang D, Cao Z (2013) Classification of bioinformatics dataset using finite impulse response extreme learning machine for cancer diagnosis. Neural Comput Appl 22 (3):457–468
40. Liu H, Liu L, Zhang H (2010) Ensemble gene selection by grouping for microarray data classification. J Biomed Inform 43(1):81–87
41. Nanni L, Lumini A (2011) Wavelet selection for disease classification by DNA microarray data. Expert Syst Appl 38(1):990–995
42. Porto-Díaz I, Bolón-Canedo V, Alonso-Betanzos A, Fontenla-Romero O (2011) A study of performance on microarray data sets for a classifier based on information theoretic learning. Neural Netw 24(8):888–896
43. Reboiro-Jato M, Díaz F, Glez-Peña D, Fdez-Riverola F (2014) A novel ensemble of classifiers that use biological relevant gene sets for microarray classification. Appl Soft Comput 17:117–126
44. Shah M, Marchand M, Corbeil J (2012) Feature selection with conjunctions of decision stumps and learning from microarray data. IEEE Trans Pattern Anal Mach Intell 34 (1):174–186
45. Sharbaf FV, Mosafer S, Moattar MH (2016) A hybrid gene selection approach for microarray data classification using cellular learning automata and ant colony optimization. Genomics 107(6):231–238
46. Zainuddin Z, Ong P (2011) Reliable multiclass cancer classification of microarray gene expression profiles using an improved wavelet neural network. Expert Syst Appl 38 (11):13711–13722
47. Dashtban M, Balafar M (2017) Gene selection for microarray cancer classification using a new evolutionary method employing artificial intelligence concepts. Genomics 109 (2):91–107
48. Ganesh Kumar P, Aruldoss Albert Victoire T, Renukadevi P, Devaraj D (2012) Design of fuzzy expert system for microarray data classification using a novel genetic swarm algorithm. Expert Syst Appl 39(2):1811–1821
49. Leung Y, Hung Y (2010) A multiple-filter-multiple-wrapper approach to gene selection and microarray data classification. IEEE/ACM Trans Comput Biol Bioinform 7 (1):108–117
50. Li HD, Liang YZ, Xu QS, Cao DS, Tan BB, Deng BC, Lin CC (2011) Recipe for uncovering predictive genes using support vector machines based on model population analysis. IEEE/ACM Trans Comput Biol Bioinform 8 (6):1633–1641
51. Liu HC, Peng PC, Hsieh TC, Yeh TC, Lin CJ, Chen CY, Hou JY, Shih LY, Liang DC (2013) Comparison of feature selection methods for cross-laboratory microarray analysis. IEEE/ACM Trans Comput Biol Bioinform 10 (3):593–604
52. Maji P (2011) Fuzzy-rough supervised attribute clustering algorithm and classification of microarray data. IEEE Trans Syst Man Cybern B Cybern 41(1):222–233
53. Maldonado S, Weber R, Famili F (2014) Feature selection for high-dimensional class-imbalanced data sets using support vector machines. Inf Sci 286:228–246
54. Nguyen T, Khosravi A, Creighton D, Nahavandi S (2015) A novel aggregate gene selection method for microarray data classification. Pattern Recogn Lett 60–61:16–23

55. Orsenigo C, Vercellis C (2012) An effective double-bounded tree-connected Isomap algorithm for microarray data classification. Pattern Recogn Lett 33(1):9–16
56. Tong M, Liu K-H, Xu C, Ju W (2013) An ensemble of SVM classifiers based on gene pairs. Comput Biol Med 43(6):729–737
57. Wang X, Park T, Carriere KC (2010) Variable selection via combined penalization for high-dimensional data analysis. Comput Stat Data Anal 54(10):2230–2243
58. Castaño A, Fernández-Navarro F, Hervás-Martínez C, Gutiérrez PA (2011) Neurologistic models based on evolutionary generalized radial basis function for the microarray gene expression classification problem. Neural Process Lett 34(2):117–131
59. Hernández-Lobato D, Hernández-Lobato JM, Suárez A (2010) Expectation propagation for microarray data classification. Pattern Recogn Lett 31(12):1618–1626
60. Lee C-P, Lin W-S, Chen Y-M, Kuo B-J (2011) Gene selection and sample classification on microarray data based on adaptive genetic algorithm/k-nearest neighbor method. Expert Syst Appl 38(5):4661–4667
61. Li J, Jia Y, Li W (2011) Adaptive huberized support vector machine and its application to microarray classification. Neural Comput Appl 20(1):123–132
62. De Paz JF, Bajo J, Vera V, Corchado JM (2011) Microcbr: a case-based reasoning architecture for the classification of microarray data. Appl Soft Comput 11(8):4496–4507
63. Ocampo-Vega R, Sanchez-Ante G, de Luna MA, Vega R, Falcón-Morales LE, Sossa H (2016) Improving pattern classification of DNA microarray data by using PCA and logistic regression. Intell Data Anal 20(s1):S53–S67
64. Twala B, Phorah M (2010) Predicting incomplete gene microarray data with the use of supervised learning algorithms. Pattern Recogn Lett 31(13):2061–2069
65. Wang H, Jing X, Niu B (2017) A discrete bacterial algorithm for feature selection in classification of microarray gene expression cancer data. Knowl-Based Syst 126:8–19
66. Zhou P, Hu X, Li P, Wu X (2017) Online feature selection for high-dimensional class-imbalanced data. Knowl-Based Syst 136:187–199
67. Cheng Q (2010) A sparse learning machine for high-dimensional data with application to microarray gene analysis. IEEE/ACM Trans Comput Biol Bioinform 7(4):636–646
68. Garro BA, Rodríguez K, Vázquez RA (2016) Classification of DNA microarrays using artificial neural networks and ABC algorithm. Appl Soft Comput 38:548–560
69. Ghaddar B, Naoum-Sawaya J (2018) High dimensional data classification and feature selection using support vector machines. Eur J Oper Res 265(3):993–1004
70. Bielza C, Robles V, Larrañaga P (2011) Regularized logistic regression without a penalty term: an application to cancer classification with microarray data. Expert Syst Appl 38 (5):5110–5118
71. Luque-Baena RM, Urda D, Gonzalo Claros M, Franco L, Jerez JM (2014) Robust gene signatures from microarray data using genetic algorithms enriched with biological pathway keywords. J Biomed Inform 49(C):32–44
72. Fernández-Navarro F, Hervás-Martínez C, Ruiz R, Riquelme JC (2012) Evolutionary generalized radial basis function neural networks for improving prediction accuracy in gene classification using feature selection. Appl Soft Comput 12(6):1787–1800
73. Wu MY, Dai DQ, Shi Y, Yan H, Zhang XF (2012) Biomarker identification and cancer classification based on microarray data using Laplace naive Bayes model with mean shrinkage. IEEE/ACM Trans Comput Biol Bioinform 9(6):1649–1662
74. Park CH, Kim SB (2015) Sequential random k-nearest neighbor feature selection for high-dimensional data. Expert Syst Appl 42 (5):2336–2342
75. Alonso-González CJ, Moro-Sancho QI, Simon-Hurtado A, Varela-Arrabal R (2012) Microarray gene expression classification with few genes: criteria to combine attribute selection and classification methods. Expert Syst Appl 39(8):7270–7280
76. Chakraborty D, Maulik U (2014) Identifying cancer biomarkers from microarray data using feature selection and semisupervised learning. IEEE J Translat Eng Health Med 2:1–11
77. Debnath R, Kurita T (2010) An evolutionary approach for gene selection and classification of microarray data based on SVM error-bound theories. Biosystems 100(1):39–46
78. García V, Sánchez JS (2015) Mapping microarray gene expression data into dissimilarity spaces for tumor classification. Inf Sci 294:362–375
79. Tabakhi S, Najafi A, Ranjbar R, Moradi P (2015) Gene selection for microarray data classification using a novel ant colony optimization. Neurocomputing 168:1024–1036
80. Lu H, Chen J, Yan K, Jin Q, Xue Y, Gao Z (2017) A hybrid feature selection algorithm for

gene expression data classification. Neurocomputing 256:56–62

81. Fan L, Poh K-L, Zhou P (2010) Partition-conditional ICA for Bayesian classification of microarray data. Expert Syst Appl 37 (12):8188–8192
82. Wang A, An N, Chen G, Li L, Alterovitz G (2015) Improving PLS-RFE based gene selection for microarray data classification. Comput Biol Med 62:14–24
83. Kumar M, Rath SK (2015) Classification of microarray using MapReduce based proximal support vector machine classifier. Knowl-Based Syst 89(C):584–602
84. Zintzaras E, Kowald A (2010) Forest classification trees and forest support vector machines algorithms: demonstration using microarray data. Comput Biol Med 40(5):519–524

# Chapter 9

# Microarray Data Normalization and Robust Detection of Rhythmic Features

**Yolanda Larriba, Cristina Rueda, Miguel A. Fernández, and Shyamal D. Peddada**

## Abstract

Data derived from microarray technologies are generally subject to various sources of noise and accordingly the raw data are pre-processed before formally analysed. Data normalization is a key pre-processing step when dealing with microarray experiments, such as circadian gene-expressions, since it removes systematic variations across arrays. A wide variety of normalization methods are available in the literature. However, from our experience in the study of rhythmic expression patterns in oscillatory systems (e.g. cell-cycle, circadian clock), the choice of the normalization method may substantially impair the identification of rhythmic genes. Hence, the identification of a gene as rhythmic could be just as an artefact of how the data were normalized. Yet, gene rhythmicity detection is crucial in modern toxicological and pharmacological studies, thus a procedure to truly identify rhythmic genes that are robust to the choice of a normalization method is required.

To perform the task of detecting rhythmic features, we propose a rhythmicity measure based on bootstrap methodology to robustly identify rhythmic genes in oscillatory systems. Although our methodology can be extended to any high-throughput experiment, in this chapter, we illustrate how to apply it to a publicly available circadian clock microarray gene-expression data and give full details (both statistical and computational) so that the methodology can be used in an easy way. We will show that the choice of normalization method has very little effect on the proposed methodology since the results derived from the bootstrap-based rhythmicity measure are highly rank correlated for any pair of normalization methods considered. This suggests, on the one hand, that the rhythmicity measure proposed is robust to the choice of the normalization method, and on the other hand, that gene rhythmicity detected using this measure is potentially not a mere artefact of the normalization method used. In this way the researcher using this methodology will be protected against the possible effect of different normalizations, as the conclusions obtained will not depend so strongly on them. Additionally, the described bootstrap methodology can also be employed as a tool to simulate gene-expression participating in an oscillatory system from a reference data set.

**Key words** Microarray, Normalization, Pre-processing, High-throughput technologies, Rhythmicity, Oscillatory systems, Circadian genes

Verónica Bolón-Canedo and Amparo Alonso-Betanzos (eds.), *Microarray Bioinformatics*, Methods in Molecular Biology, vol. 1986, https://doi.org/10.1007/978-1-4939-9442-7_9, 

## 1 Introduction

Microarray gene-expression analysis is hampered by the noisy nature of the data [1, 2] that is intrinsic to each array. Hence, a previous gene-expression data pre-processing is required to remove (or reduce) sources of variation of non-biological origin among arrays [3, 4]. Several pre-processing methods are available in the literature, such as the model-based expression index (MBEI) [5], MAS 5.0 [6, 7] and robust multi-array average (RMA) [8]. The pre-processing goal is twofold, first to reduce non-biological variability, and second to take data from a tri-dimensional array of raw intensities (i.e. probe level) to a gene-expression matrix (i.e. gene level). Pre-processing methods usually involve three distinct steps called *background correction, normalization*, and *summarization* [9]. Normalization step plays a key role in pre-processing [3, 10], since it removes technical (i.e. non-biological) variations from the expression data. A variety of strategies are available in the literature to normalize gene-expression data, and in this chapter we focus on the following popular ones: *Quantile* [3], *(Cyclic) Loess* [3], *Contrast* [11], *Invariant Set* [5], *Qspline* [12], *and variance stabilization normalization (VSN)* [13]. However, each strategy is based on specific model and assumptions and consequently, the resulting normalized expression data, and the downstream analyses, are likely to depend upon the normalization method chosen (*see* Fig. 1). One may refer to [14] for details.

After pre-processing, one of the most important problems in practice regarding gene-expression data from biological processes (c.f. metabolic cycle [15], cell-cycle [16–19] or circadian clock [20, 21]) is to detect the components (genes) that govern those oscillatory systems exhibiting rhythmic or periodic patterns over time. There exists in the literature a wide variety of procedures to detect rhythmicity including among other, those based on autocorrelation [22], cosine curve-fitting [23], or non-parametric tests like JTK_Cycle [24] and RAIN [25]. Recently, [21] proposed ORIOS, an algorithm based on order restricted inference to detect and classify rhythmic genes in oscillatory systems. Order restricted inference incorporates the prior knowledge that certain parameters satisfy an order restriction into the analysis improving the performance of the statistical procedures, *see* [26]. It has proved to be quite useful, for example, in the study of biologically interesting problems in the cell-cycle among many others. Applications of interest in this and other related fields together with the appropriate software can be found in [27–32].

In order to make a clear exposition of our rhythmicity detection methodology, in this chapter we will focus on rhythmicity detection of circadian gene-expression and we will consider ORIOS as detection algorithm. As in any other detection algorithm (*see* [14]) the

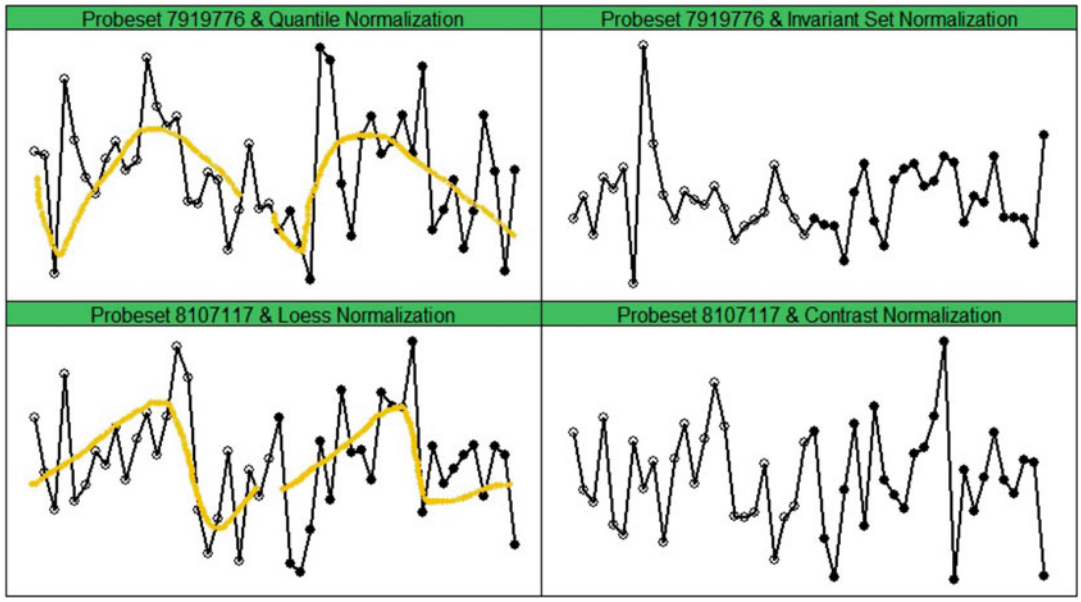


**Fig. 1** Time-course gene-expression for Probeset 7919776 (top) and 8107117 (bottom). Both probesets are identified as rhythmic by ORIOS according to *Quantile* and *Loess* normalizations, respectively, but they are identify as non-rhythmic by ORIOS for *Invariant Set* and *Contrast* normalizations, respectively

results from ORIOS substantially depend upon, among other factors, the normalization strategy chosen. This fact is reflected in meaningful differences in the number of genes identified as rhythmic. For instance, the percentage of genes identified as rhythmic in the human cell lines microarray experiment U2OS, analysed in this chapter (*see* Subheading 2 for details on the data set), decreases more than a 40% when VSN normalization is chosen instead of Loess. These differences can be also illustrated graphically. Figure 1 shows time-course data on two *probesets* (genes) from U2OS, namely Probeset 7919776 and Probeset 8107117 (from now on we refer to them as *ProbesetA* and *ProbesetB*, respectively). Data are normalized using *Quantile, Invariant Set, Loess*, and *Contrast* strategies. The top panel of Fig. 1 provides the time-course plots of the *ProbesetA* using *Quantile* (left panel) and *Invariant Set* (right panel) normalization methods. The bottom panel of Fig. 1 displays the time-course plots of the *ProbesetB* using *Loess* (left panel) and *Contrast* (right panel) normalization strategies. As can be seen, the gene-expression profiles are markedly different regarding the normalization method employed. Moreover, *ProbesetA* and *ProbesetB* are detected as rhythmic genes by ORIOS if *Quantile* or *Loess* normalizations are used, respectively, but they are declared to be non-rhythmic genes if *Invariant Set* or *Contrast* strategies are employed. This observation that normalization method may impact the rhythmicity of a gene is not limited to the above genes

but is rather a common feature of time-course data as noted in Fig. 4.

In a recent paper [14], we introduced a bootstrap-based rhythmicity measure that is strongly correlated across the normalization strategies. In this chapter we review that methodology and provide step-by-step details for its application, so that an interested researcher can apply it in a smooth way and obtain a set of rhythmically ranked genes from an appropriate data set. As a product of the methodology the genes declared as rhythmic and appearing in the top positions of rhythmicity ranks under one normalization are expected to also appear in top positions under a different one thus reinforcing the conclusions that may be obtained.

In addition, a by-product of the proposed bootstrap methodology is that it can be used for simulating potentially realistic microarray experiments. Despite that several authors have provided algorithms for simulating gene-expression microarray data [33–36], many of them are not broadly applicable, since they are designed to cover specific requirements. Unlike those procedures, the proposed algorithm is very generic and allows us to simulate microarray gene experiment from probe to gene level from a microarray reference data set.

The layout of the chapter is as follows. In Subheading 2 we start describing the data set and the software employed to pre-process microarray data. Subheading 3 details the bootstrap methodology to define a robust rhythmicity measure and includes a brief remark about its usage to simulate microarray experiments. In this section, we also provide the results from applying the bootstrap methodology on the mentioned data set, showing the good performance of the proposed rhythmicity measure and how the improvement on rank correlations among the different normalizations can be useful in biological practice. In Subheading 4, short step-by- step instructions to generate microarray data sets are provided and some important issues concerning both methodology and data analysis are highlighted.

## 2 Materials

We analyse a publicly available time-course gene-expression data from the U2OS human cell lines, a conventional model of the autonomous circadian clock [37, 38]. The human osteosarcoma U2OS cell line is one of the first generated cell lines and is used in various areas of biomedical research, since it is a primary malignant tumour of the bone affecting children and young adults (age 15–29 years), as well as adults in later life (age $>$ 60 years), *see* [39]. Data are online available at NCBI GEO (http://www.ncbi.nlm.nih.gov/geo/) with GSE13949 GEO accession number and they have been widely analysed in the literature to identify rhythmicity, see among

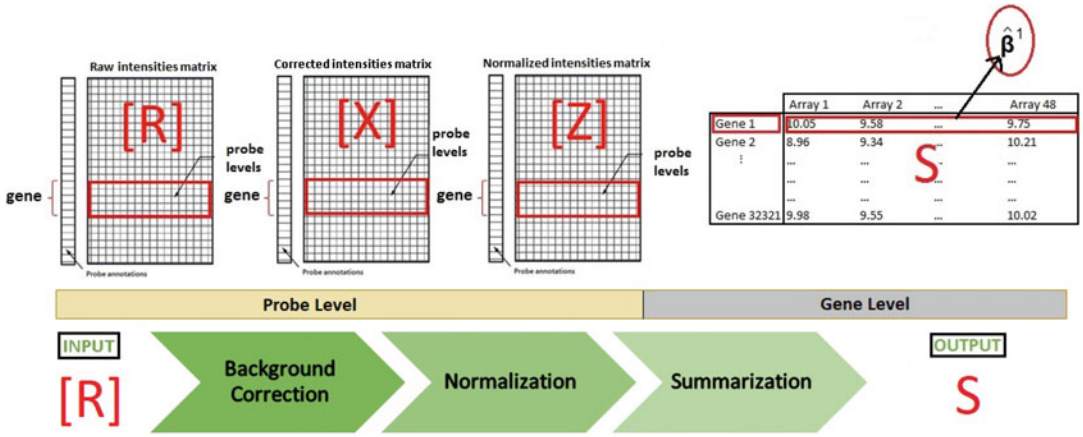


**Fig. 2** Work-flow of pre-processing procedure from raw intensities **R** to gene-expression matrix **S**

others [14, 20, 21, 24]. Note 1 in Subheading 4 provides additional details to access and download the microarray data.

Raw data are expressed at probe level. Each gene $g$, $g = 1, \ldots, G$, is associated with a $P \times T$ matrix of gene-intensity values, where $P$ and $T$ represent the number of probes and arrays (time points) regarding the gene $g$, respectively, and $G$ denotes the number of genes in the data set. The number of probes $P$ varies from one gene to other, whereas the number of time points $T$ is usually fixed. Data must be pre-processed not only to reduce variability but also to obtain a $G \times T$ matrix of gene-expression values. A scheme of pre-processing procedure is illustrated in Fig. 2. Note 2 in Subheading 4 illustrates a step-by-step guide to pre-process microarray data.

In this chapter, raw data are pre-processed according to RMA procedure, following the lines described in [8]. The six normalization methods mentioned before are implemented together with the background correction and summarization steps proposed in [8]. To do so, we make use of the R-packages Affy [40] and VSN [13] from the Bioconductor Project [41] as is shown in Note 2 in Subheading 4. Resulting from pre-processing, the U2OS data set consists of 32321 single gene-expressions obtained along 48 time points representing two periods of data of 24 h length.

## 3 Methods

In this section we provide a standard rhythmicity measure that allows to declare genes as rhythmic or not. In addition, for the sake of completeness and self-containment, we review the bootstrap-based methodology appearing in [14] both to robustly identify rhythmicity and to simulate microarray data sets. Finally,

we illustrate the performance of bootstrap-based methodology on improving rank correlations in the U2OS cell lines data set.

The methodology described in this chapter considers ORIOS as rhythmicity detection algorithm and six different normalization methods, called *(1) Quantile, (2) (Cyclic) Loess, (3) Contrast, (4) Invariant Set, (5) Qspline, (6) VSN*. As already mentioned, it can be extended to other normalization methods and/or detection algorithms, *see* Note I in Subheading 4.

### 3.1 Standard Measure of Gene Rhythmicity

The usual way to detect gene rhythmicity, not only in ORIOS but also in the most popular detection algorithms (*see* [21, 24, 25]), is to perform a set of hypotheses tests as the following one for the genes in the data set:

$$\begin{aligned} H_0 &: \text{ Theunderlyinggenepatternisflat} \\ H_1 &: \text{ Theunderlyinggenepatternisrhythmic} \end{aligned} \tag{1}$$

Then, rhythmicity detection is given in terms of the Benjamini–-Hochberg (BH) [42] adjusted $p$-values derived from (1). BH adjusted $p$-values are then used to define a standard measure of gene rhythmicity as follows. Let $p\text{-value}^{g}(n)$ denote the BH adjusted $p$-value of the gene $g$ with respect to the normalization method $n$, for $n = 1, \ldots, 6$ and $g = 1, \ldots, G$. The standard measure of gene rhythmicity associated with $g$ (for $g = 1, \ldots, G$) is defined as:

$$M^{g}(n) = 1 - p\text{–value}^{g}(n). \tag{2}$$

The components of vector $\mathbf{M}(n) = (M^{1}(n), \ldots, M^{G}(n))'$ take values between 0 and 1. Measurements close to 0 indicate potentially non-rhythmic genes and close to 1 indicate potentially rhythmic genes. For instance, for *ProbesetA* and *ProbesetB* involved in Fig. 1 we have $M^{ProbesetA}(Quantile) = 0.989$, $M^{ProbesetA}(InvariantSet) = 0.871$, $M^{ProbesetB}(Loess) = 0.988$, and $M^{ProbesetB}(Contrast) = 0.891$. Taking $\mathbf{M}(n) \geq 0.95$, as the rhythmicity criterion, those numbers reinforce, as was illustrated graphically (*see* Fig. 1), that both probesets are potentially rhythmic under *Quantile* and *Loess* normalization methods, but not under *Invariant Set* and *Contrast*, respectively.

### 3.2 Bootstrap Methodology

In the following we review the bootstrap methodology proposed in [14] in order to introduce $\mathbf{M}_{\text{Robust}}(n)$, a more robust measurement with respect to the normalization method. We also give details on how to use the bootstrap methodology to simulate microarray gene-expression data from a reference data set.

Let $\mathbf{R}$ be a reference microarray data set of raw intensity values derived from an oscillatory system, *see* Fig. 2. We propose a parametric bootstrap [43] based on a linear model from the tri-dimensional array of corrected intensities $\mathbf{X}$ derived from the reference data set.

Let $X_{pt}^{g}$ denote the corrected intensity value for gene $g$ on probe $p$ at time point $t$ after background correction, where $g = 1, \ldots, G$, $p = 1, \ldots, P$, and $t = 1, \ldots, T$. For each bootstrap replication $b = 1, \ldots, B$, simulated intensity data sets $\mathbf{X}^{(b)*}$ are generated as follows:

$$\log_2(X_{pt}^{(b)g*}) = \hat{\alpha}_{p}g + \hat{\beta}_{t}g + \varepsilon_{pt}^{(b)g*}, \tag{3}$$

where $g = 1, \ldots, G$, $p = 1, \ldots, P$, $t = 1, \ldots, T$, $b = 1, \ldots B$. $\{\hat{\alpha}_{p}g\}_{g=1}^{G}$, and $\{\beta_{t}\hat{g}\}_{g=1}^{G}$ denote original estimates of probe and array effects, respectively, obtained from corrected (but non-normalized) intensities $\mathbf{X}$ in the pre-processing of the reference data $\mathbf{R}$, *see* Fig. 2 for details. $\varepsilon_{pt}^{(b)g*}$ are identically and independently distributed according to a normal distribution $N(0, \sigma\,\hat{2}g)$, where $\sigma\,\hat{2}g$ is the usual mean squared error (*MSE*) under the original two-way model. *See* Note II in Subheading 4 for details.

The values $\mathbf{X}^{(b)*}$ are expressed at probe level, thus normalization and summarization steps must be conducted (*see* Note 2 in Subheading 4). $\mathbf{X}^{(b)*}$ are normalized regarding the six normalization strategies considered in this chapter. Summarization step follows the methodology described in [8]. There, a median polish algorithm is used to estimate model parameters [44]. This algorithm takes into account probe and array effects, similar to a two-way ANOVA-based estimation procedure, except that it employs medians instead of means to ensure robustness to outliers. As a result, for each bootstrap replication $b\,(b = 1, \ldots, B)$, a $G \times T$ matrix $\mathbf{S}^{(b)*}$ of simulated gene-expression is obtained as output of the pre-processing procedure. Thus, in addition to detect rhythmic genes robustly, our bootstrap methodology may also be used as a tool to generate reasonably realistic gene-expression data (i.e. $\mathbf{S}^{(b)*}$). Note 3 in Subheading 4 gives full details to conduct bootstrapping in order to simulate microarray data from a reference data set.

Figure 3 illustrates the performance of our methodology as a microarray simulator. Time-course gene-expression data of gene Per3 and Probeset 7892507 are displayed on top and bottom left panels of Fig. 3, respectively. Figure 3 shows how well our bootstrap-based simulated data (right panels in Fig. 3) resembles the pattern of expression of the real data (left panels), both for rhythmic (gene Per3) or non-rhythmic (Probeset 7892507) genes. The reason for this good behaviour is that in Eq. 3, we just bootstrap the residuals. Thus the mean signal over the bootstrap samples retains the original expression and hence there is no loss of information in the mean signal through bootstrapping. Although in this chapter microarray simulation is addressed to detect rhythmic features, our methodology can be adapted to other biological applications. It is important to note that, in practice, microarray simulation may be a hard task in terms of storage and memory

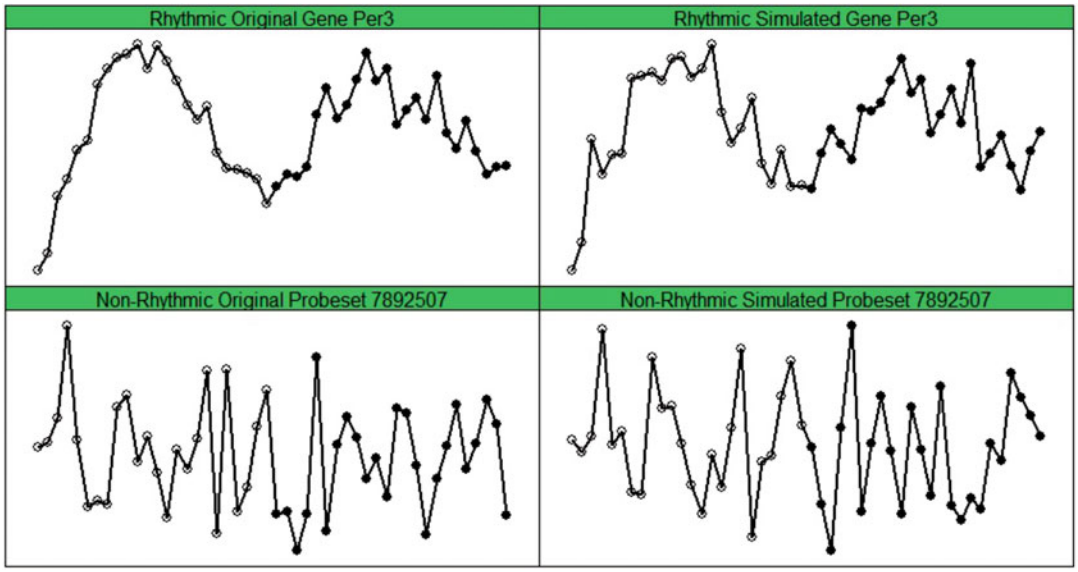


**Fig. 3** Original vs simulated gene-expression attained after bootstrapping from U2OS using *Quantile* normalization. Top: Original (left) and simulated (right) time-course gene patterns of the rhythmic gene Per3. Bottom: Original (left) and simulated (right) time-course gene patterns of the non-rhythmic Probeset 7892507

computation since data derived from bootstrap procedure are expressed at probe level. *See* Note III in Subheading 4 for further details.

### 3.3 Robust Measure of Gene Rhythmicity

Given a normalization strategy $n$ and a random realization of the data, the above bootstrap methodology allows us to provide the gene rhythmicity measure $\mathbf{M}_{\text{Robust}}(n)$ that reduces the impact of the normalization method chosen to detect rhythmicity.

Our parameter of interest is $\boldsymbol{\theta}(n) = \mathbb{E}(\mathbf{M}(n))$, the vector containing the true rhythmicity values of the genes under normalization strategy $n$. $\boldsymbol{\theta}(n) = \mathbf{M}(n)$ is an estimator of this value, where $\mathbf{M}(n)$ is the rhythmicity statistic defined in Eq. 2. Now, we define a new (and better) estimator for our parameter of interest using the bootstrap samples as follows. For each bootstrap sample $b$ ($b = 1, 2, \ldots, B$), we define $\boldsymbol{\theta}(\hat{b})*(n)$ as the bootstrap estimate of $\boldsymbol{\theta}(n)$. It is determined computing the statistic (Eq. 2) on the simulated gene-expression matrix $\mathbf{S}^{(b)*}$ derived from Eq. 3. Now, the robust gene rhythmicity measure $\mathbf{M}_{\text{Robust}}(n)$ is defined using the bootstrap samples mean and a measure of sample to sample variation as:

$$\mathbf{M}_{\text{Robust}}(n) = \hat{\mathbb{E}}(\hat{\boldsymbol{\theta}}(n)) - \widehat{\text{RMS}}(\hat{\boldsymbol{\theta}}(n)), \tag{4}$$

where $\hat{\mathbb{E}}(\hat{\boldsymbol{\theta}}(n))$ and $\widehat{RMSE}(\hat{\boldsymbol{\theta}}(n))$ are defined as:

$$\hat{\mathbb{E}}\left(\boldsymbol{\theta}(n)\right) = \frac{1}{B}\sum_{b=1}^{B}\hat{(\boldsymbol{\theta}}\left(\hat{b}\right)*(n)) \tag{5}$$

$$RMSE\left(\hat{\boldsymbol{\theta}}(n)\right) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\boldsymbol{\theta}\left(\hat{b}\right)*(n) - \hat{\boldsymbol{\theta}}(n))^2}. \tag{6}$$

This gene rhythmicity measure is termed as "robust" since, as demonstrated later in the chapter, it reduces the effect of the normalization method used correcting for sample to sample variation in the rhythmicity measure (i.e. *RMSE*).

### 3.4 Results

Now we apply our methodology on the data set U2OS, consisted of 32321 gene-expressions along 48 time points. Expression data were pre-processed according to the six normalization methods mentioned above. For each normalization method $n$, ORIOS algorithm is used to detect rhythmicity, taking $M^g(n) \geq 0.95$ (resp. $M^g_{\text{Robust}}(n) \geq 0.95$) as the criterion to declare a gene to be (resp. robustly) rhythmic, for $n = 1, \ldots, 6$. (*See* note IV in Subheading 4 for a discussion about the choice of that threshold.) Figure 4 clearly shows how the normalization choice presents a large impact on the number of genes declared to be rhythmic by the standard measure.

To illustrate how our bootstrap-based gene rhythmicity measure, $\mathbf{M}_{\text{Robust}}$, increases the correlation among different

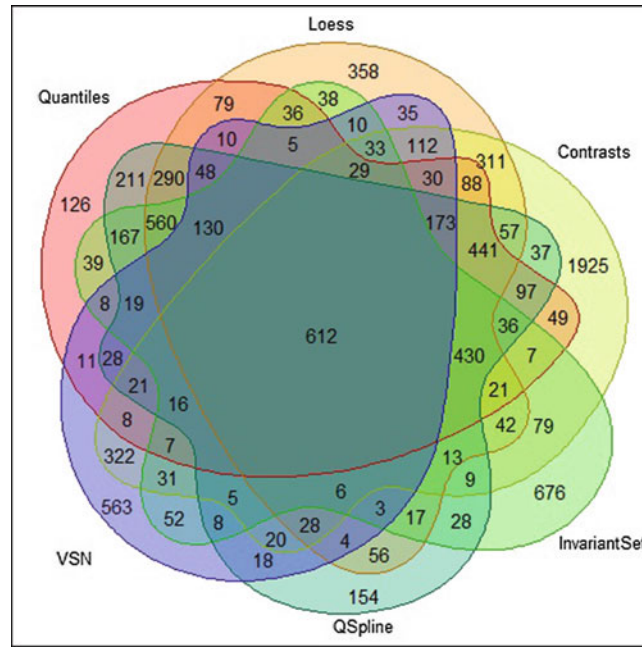


**Fig. 4** Joint rhythmic gene distribution in U2OS across normalization strategies, according to $M^g(n) \geq 0.95$, for $n = 1, \ldots, 6$ and $g = 1, \ldots, 32321$

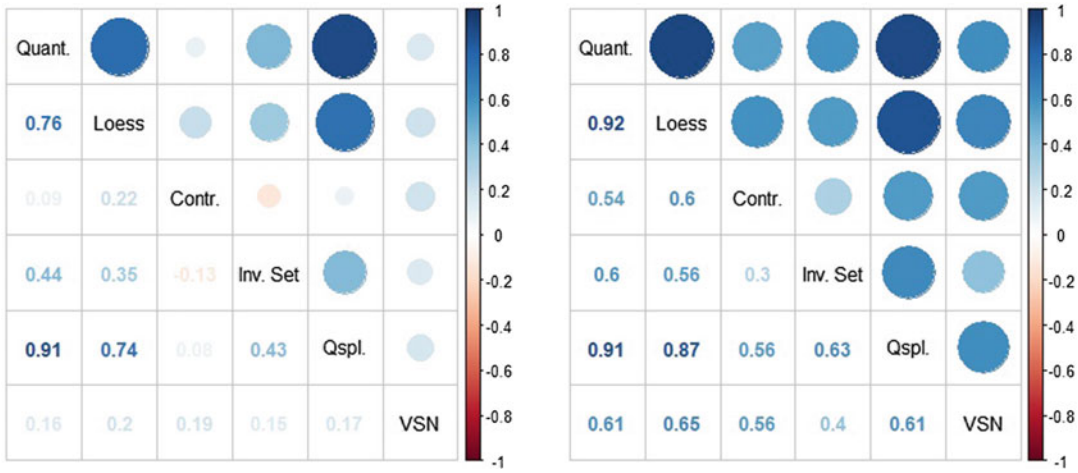


**Fig. 5** Spearman rank correlation coefficients between all pairs of normalization procedures considering the standard measure of rhythmicity **M** (left) and the proposed robust measure $\mathbf{M}_{\text{Robust}}$ (right) for the ORIOS algorithm using the 8882 genes, showing a highly increased consistency due to bootstrapping

normalization methods, we compute the Spearman rank correlation coefficients between $\mathbf{M}_{\text{Robust}}(n_i)$ and $\mathbf{M}_{\text{Robust}}(n_j)$ for all pairs of normalization methods $n_i$, $n_j$, $i \neq j$. The correlation coefficients obtained are then compared with those corresponding to the standard measure. To reinforce this fact, we also compute the per cent of concordance of rhythmic and non-rhythmic genes across all normalization methods comparing standard and robust measures. Correlation (Fig. 5) and concordance (Fig. 6) analyses provided in this chapter are limited to the set of 8882 genes in U2OS that were declared to be rhythmic by at least one of the normalization strategies $n$, according to the criterion $M^g(n) \geq 0.95$, for $n = 1, \ldots, 6$ and $g = 1, \ldots, 32321$. For every pair of normalization methods, Figs. 5 and 6 display a substantial increase from the left to the right panel. In both figures, left-hand panels are related to $\mathbf{M}(n)$, whereas the right ones correspond to $\mathbf{M}_{\text{Robust}}(n)$. To better illustrate this fact, we consider the Spearman correlation coefficient between $\mathbf{M}(Loess)$ and $\mathbf{M}(VSN)$. We observe that it increases from 0.2 (left panel of Fig. 5) to 0.65 (right panel of Fig. 5) when $\mathbf{M}_{\text{Robust}}(Loess)$ and $\mathbf{M}_{\text{Robust}}(VSN)$ are considered, which is a substantial increase. The increase is even more dramatic when considering the percentage of concordant genes. In this case, it increases dramatically by more than 44%, from 55.26 to 99.69% (*see* Fig. 6). Moreover, that increase can be further illustrated graphically. Figure 7 displays the scatter plot of the normalization pairs *Loess* and *VSN* regarding standard (left panel) and robust (right panel) rhythmicity measures. Left panel of Fig. 7 presents a highly non-elliptic scatter of points with no clear correlation, whereas the scatter plot on the right panel appears to be very elliptic with smaller minor axis. As a by-product, concordance (Fig. 5) and correlation (Fig. 6) analyses together

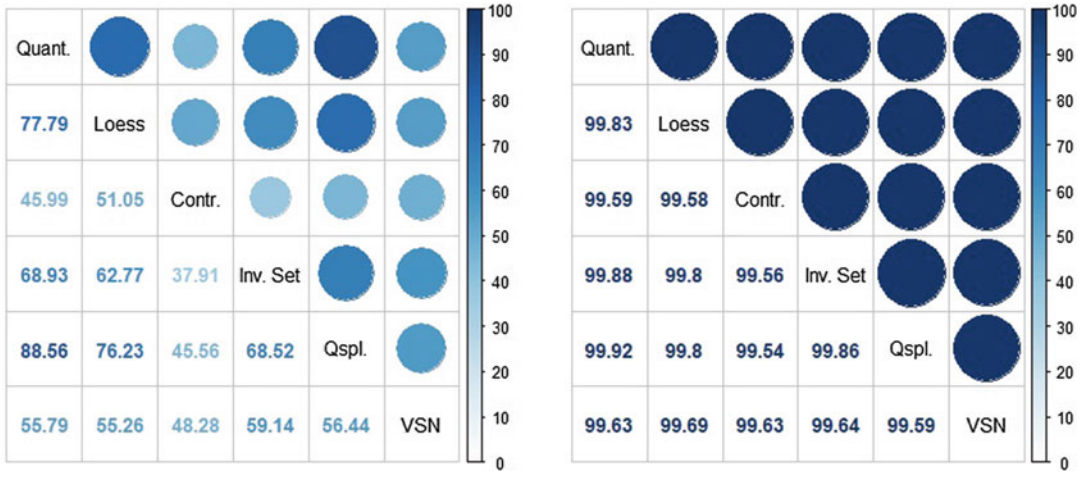


**Fig. 6** Percentage of (rhythmic and non-rhythmic) concordant genes before (left) and after (right) bootstrapping for all pairs of normalization procedures using the 8882 genes. Bootstrapping increases significantly the concordance

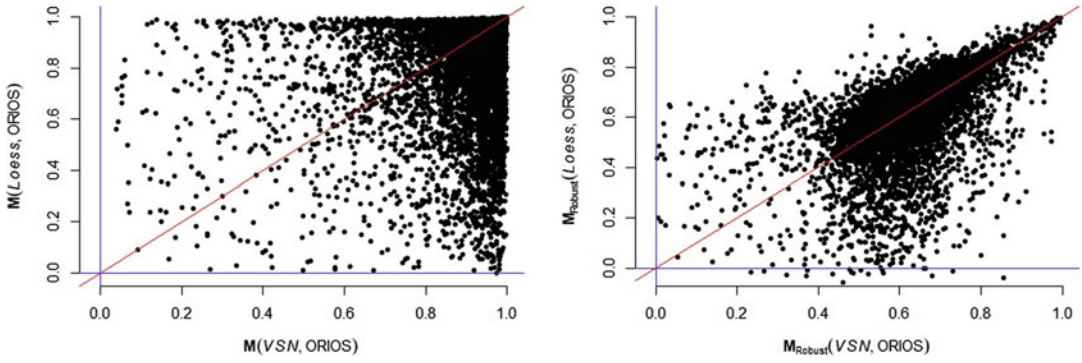


**Fig. 7** Pairwise scatter plots of $(M^g(Loess), M^g(VSN))$ (left) and $(M^g_{\text{Robust}}(Loess), M^g_{\text{Robust}}(VSN))$ (right). Red line is the 45° diagonal and the blue lines are the Cartesian axes. Right side scatter plot shows a much more elliptical shape and a higher correlation indicating higher consistency between this pair of normalizations

with Fig. 4 imply that the *Contrast* normalization method may be the least preferred method among the six normalization schemes, as the corresponding robust measure seems to be least correlated with the others and also because the number of genes exclusively declared to be rhythmic by that method regarding the standard measure is extremely high compared with the other strategies.

It is worthy to mention that Spearman correlation coefficients are based on ranks. Thus, the increase in the right-hand panel of Fig. 5 suggests that gene rhythmicity ranks are correlated across the normalization methods considered here when our bootstrap methodology is applied. As a consequence, a gene declared to be rhythmic (resp. non-rhythmic) under one normalization scheme is likely to be

**Table 1**
**Rank rhythmicity position for the genes *Gene1*,. . .,*Gene10* regarding standard (left side) and robust (right side) rhythmicity measures**

| | M(*n*) | | | | | | $M_{Robust}(n)$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Labels** | ***Quant.*** | ***Loess*** | ***Contr.*** | ***Inv. Set*** | ***Qspl.*** | ***VSN*** | ***Quant.*** | ***Loess*** | ***Contr.*** | ***Inv. Set*** | ***Qspl.*** | ***VSN*** |
| *Gene1* | 133 | 106 | 283 | 98 | 35 | 15 | 7 | 6 | 6 | 9 | 6 | 6 |
| *Gene2* | 9 | 4 | 6 | 11 | 14 | 107 | 3 | 3 | 4 | 3 | 2 | 11 |
| *Gene3* | 2 | 1 | 1 | 1 | 2 | 3 | 1 | 1 | 1 | 1 | 1 | 1 |
| *Gene4* | 105 | 139 | 348 | 101 | 102 | 57 | 27 | 24 | 18 | 25 | 29 | 24 |
| *Gene5* | 1175 | 1382 | 2373 | 1019 | 1181 | 1146 | 15 | 18 | 17 | 11 | 15 | 15 |
| *Gene6* | 749 | 360 | 357 | 637 | 781 | 381 | 6 | 8 | 8 | 23 | 7 | 5 |
| *Gene7* | 1153 | 1311 | 1689 | 998 | 1199 | 967 | 33 | 25 | 11 | 22 | 32 | 17 |
| *Gene8* | 1460 | 818 | 360 | 1236 | 1713 | 156 | 9 | 10 | 9 | 8 | 12 | 7 |
| *Gene9* | 1 | 2 | 3 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 3 | 3 |
| *Gene10* | 276 | 299 | 98 | 236 | 526 | 53 | 4 | 5 | 3 | 5 | 5 | 4 |

rhythmic (resp. non-rhythmic) under a different one. To illustrate this point, consider again the two genes plotted in Fig. 1. As noted in Subsection 3.1, the rhythmicity calls on these two genes largely depend on the normalization method $n$, when the standard criterion $M^g(n) \geq 0.95$ is considered. Yet, under the criterion $M^g_{\text{Robust}}(n) \geq 0.95$, neither of these genes are considered to be rhythmic. In particular, according to the normalization methods used earlier in Fig. 1, we obtained the following robust rhythmicity measures $M^{ProbesetA}_{\text{Robust}}(Quantile) = 0.698$; $M^{ProbesetA}_{\text{Robust}}(InvariantSet) = 0.292$; $M^{ProbesetB}_{\text{Robust}}(Loess) = 0.696$, and $M^{ProbesetB}_{\text{Robust}}(Contrast) = 0.492$. None of these numbers exceed 0.95.

To show the practical interest of the procedure exposed and reinforce our rank analysis conclusions, let us suppose that a researcher wants to identify a set of rhythmic genes to explore more deeply. A good choice might be to consider those genes appearing in the first positions in the rhythmicity ranks under several (or all) normalization methods. If this is done considering all normalizations and using our bootstrap procedure, we will obtain the 10 genes appearing in Table 1 (*see* Note V in Subheading 4 for full details on how to obtain these genes and Table 2 for the original symbol names of these genes). We can see that, after bootstrapping, all of them appear on very good positions (right side of Table 1), while this does not happen if the standard procedure is considered (left side of Table 1), so that some of these genes might have been left out due to large variations on their rank positions.

**Table 2**
**Original symbol genes associated with the labelled genes *Gene1*,. . ., *Gene10***

| Label | Symbol |
|---|---|
| *Gene1* | Probeset 7894590 |
| *Gene2* | Gene Per3 |
| *Gene3* | Gene Arntl |
| *Gene4* | Probeset 7973867 |
| *Gene5* | Gene Pdpk1 |
| *Gene6* | Probeset 8013519 |
| *Gene7* | Gene Rnu2-1 |
| *Gene8* | Gene Ccdc74a |
| *Gene9* | Gene Nr1d2 |
| *Gene10* | Probeset 8180318 |

A more graphical example of this issue can be observed in Fig. 8, where a visually rhythmic gene (gene Nr1d1) appears on the top 5 rank for five of normalizations for the robust measure, *see* Table 3 (recall that *Contrast* normalization is one of the less correlated with the rest) while it only appears on the top 30 rank for the standard procedure on one of the normalizations. Thus, it is revealed that much more strong coherence on the rhythmicity appears after considering our bootstrap methodology.

## 4 Notes

This section provides programming instructions to simulate a microarray data set using R (*see* **Notes 1** to **3**). In addition, we discuss some important points related to methodology and data analysis (*see* **Notes I** to **V**).

Notes 1 to 3 contain short step-by-step R-code instructions to read, pre-process, and simulate an Affymetrix microarray data set using U2OS (with GSE13949 GEO accession number) as the reference microarray experiment.

1. *Reading data.*

   Obtain the .CEL extension files associated with your Affymetrix GeneChip experiment that contains the raw intensity data files resulting from array scanning process. Those files can be downloaded from databases like NCBI GEO (http://www.ncbi.nlm.nih.gov/geo/) or CircaDB (http://circadb.hogeneschlab.org/) [45]. Then, load the *Affy* package from

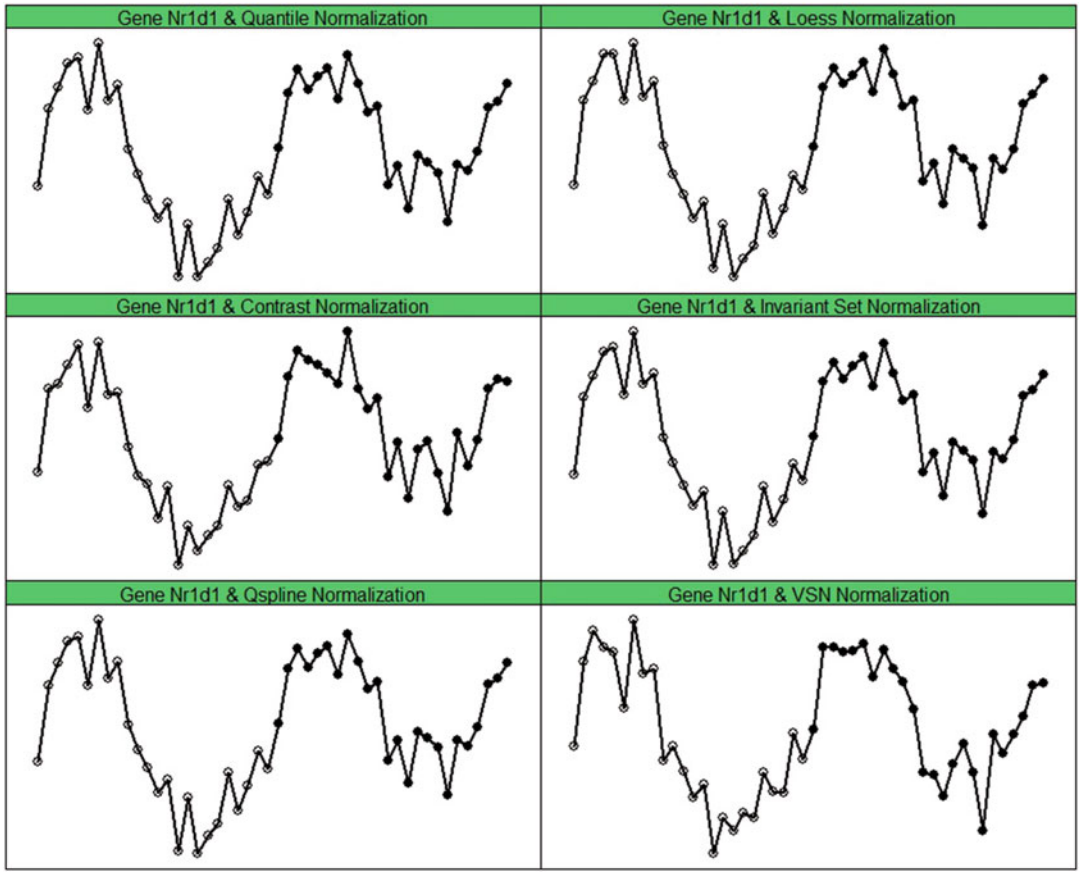


**Fig. 8** Time-course gene-expression for the gene Nr1d1 regarding the six normalization methods

**Table 3**
**Rank rhythmicity position for the gene Nr1d1 regarding standard (left side) and robust (right side) rhythmicity measures**

| | $M(n)$ | | | | | | $M_{Robust}(n)$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Gene** | ***Quant.*** | ***Loess*** | ***Contr.*** | ***Inv. Set*** | ***Qspl.*** | ***VSN*** | ***Quant.*** | ***Loess*** | ***Contr.*** | ***Inv. Set*** | ***Qspl.*** | ***VSN*** |
| Nr1d1 | 45 | 32 | 144 | 34 | 49 | 4 | 5 | 4 | 68 | 4 | 4 | 2 |

*Bioconductor* and read the .CEL data files in order to obtain raw intensity data (`dat`) which are expressed at probe level. Make sure that .CEL files are saved in your R working directory.

```
source("https://bioconductor.org/biocLite.R")
biocLite("affy")
```

```
library("affy")
dat <- ReadAffy()
```

2. *Pre-processing data.*

   Raw data are background corrected as in the RMA pre-processing method [8].

```
backDat <- bg.correct(dat, method = "rma")
```

   Then, corrected data are normalized according to one of the normalization strategies proposed in the chapter. Package *VSN* from *Bioconductor* is required to conduct *VSN* normalization.

```
normDat<-normalize.AffyBatch.quantiles(backDat,
type="pmonly")
normDat<-normalize.AffyBatch.loess(backDat,
type="pmonly")
normDat<-normalize.AffyBatch.contrasts(backDat,
type="pmonly")
normDat<-normalize.AffyBatch.invariantset(backDat,
baseline.type="median",type="pmonly")
normDat<-normalize.AffyBatch.qspline(backDat,
type="pmonly")
normDat<-justvsn(backDat)
```

   Once data have been normalized, they are summarized to obtain a matrix of gene-expression values. To do so we make use of the summarization step implemented in the RMA pre-processing method [8].

```
summDat<-exprs(rma(normDat,background=FALSE,
normalize=FALSE))
```

3. *Simulating data.*

   Before starting, check your R memory limit using `memory.limit()`. Microarray analyses involve high dimensional data to be stored, thus it is likely that you have to increase its limits using `memory.limit(size=newMemoryLi mit)`. Check `?memory.limit` to assign a value to `newMemoryLimit`.

   The simulation of microarray experiments requires, on the one hand, an array of raw intensity data (`dat`) from a reference data set. On the other hand, we also need the specific chip description file (CDF) associated with the reference experiment (`pd.hugene.1.0.st.v1` for the case of U2OS). CDF files describe the layout for Affymetrix GeneChip arrays, among others, they contain the location of the probes on the chip as well as intrinsic phenotype features of the experiment. *Bioconductor* holds packages containing the environment associated with such CDF files.

```
biocLite("pd.hugene.1.0.st.v1")
library("pd.hugene.1.0.st.v1")
```

Then, microarray simulation is conducted following the lines described in Subheading 3.2 as follows. First, the parameter estimators of the two-way model associated with the reference experiment are subtracted using the function called `modelComponentX`. Then, bootstrap replication is built as is proposed in Eq. 3 using `buildingX`. Finally, `simulPhenoData` function provides an *AffyBatch* class-object reproducing the metadata and phenotype of the reference microarray experiment. All these functions, as well as a user friendly R-code and a user guide, can be found in http://www.eio.uva.es/~miguel/robustdetectionprocedure.html.

```
originalEst <- modelComponentXPar(dat,Ngenes=32321)
X_boot <- buildingXPar(dat,originalEst,Ngenes=32321
,timePoints=48)
X_b_star <- simulPhenoData(dat,X_boot,timePoints=48)
```

Finally, simulated data set (`X_b_star`) is normalized and summarized following the instruction given above (see *preprocessing data*) to obtain a matrix of simulated gene-expression values imitating the realistic features of the reference data set. As a closing remark, since microarray experiments involve high dimensional data, parallel programming may increase computational efficiency.

Next, notes I to V highlights some important issues concerning the methodology and the data.

I. Bootstrap-based methodology introduced in the chapter is focused on six normalization strategies (*Quantile, (Cyclic) Loess, Contrast, Invariant Set, Qspline*, and *VSN*) and a rhythmicity detection algorithm (ORIOS). However, the general formulation of the proposed methodology allows us to extend it (*see* [14]) to other choices of normalization strategies and/or rhythmicity detection algorithms according to the reader necessities.

II. Error terms in Eq. 3 are i.i.d. according to a normal distribution with mean 0 and variance $\sigma\,\hat{2}g$, for $g = 1, \ldots, G$. $\sigma\,\hat{2}g$ is the MSE under the original two-way model associated with each gene. It is defined as follows:

$$\hat{\sigma}^{2g} = \frac{\sum_{p=1}^{P}\sum_{t=1}^{T} Y_{pt}^{g} - \overline{Y}_{p.}^{g} - \overline{Y}_{.t}^{g} + \overline{Y}_{..}^{g}}{(P-1)(T-1)}, \quad \text{for } g = 1 \ldots, G, \tag{7}$$

where $Y_{pt}^{g} = \log_2(X_{pt}^{g})$. The terms $\overline{Y}_{p.}^{g}$, $\overline{Y}_{.t}^{g}$, and $\overline{Y}_{..}^{g}$ denote row, column, and global means, respectively.

III. The bootstrap methodology introduced in this chapter generates microarray data expressed at probe level. Probe level analyses involve high dimensional data. For instance, in U2OS, each bootstrap replication generates a $\sim 788k \times 48$ matrix of corrected intensities. It supposes a challenge when many simulated microarray data sets regarding different normalization schemes are considered. Thus, microarray simulation is a hard procedure in terms of storage and memory consumption.

IV. One of the major difficulties dealing with microarray data is the noisy nature of the data. Normalization stage is key to address this issue reducing non-biological variations. Even so, other factors (c.f. environmental conditions, errors derived from measuring devices, or tissue specific features) might also impact on subsequent analyses such as rhythmicity detection. In addition to this, the expected rate of rhythmic genes largely varies from ones data sets to others. For instance, it is known that the rhythmicity rate in U2OS cell lines is lower ($\sim 3\%$) than it is for other data sets like mouse liver ($\sim 20\%$). One may refer to [21] for details. Thus, rhythmicity criterion thresholds must be chosen according to the above comments. Moreover, motivated by rank analyses results, authors insist on assessing relative gene rank positions, rather than fixing a cut off threshold.

V. For the U2OS experiment, we propose as a suitable set of genes to be explored more deeply the set of 10 genes which are declared to be rhythmic by all the normalization strategies according to the criterion $M^{g}_{\text{Robust}}(n) \geq 0.95$, for $n = 1, \ldots, 6$ and $g = 1, \ldots, 32321$. We refer to them as *Gene1*,. . .,*Gene10*. Table 2 states the correspondence between those labels and their corresponding gene symbols.

## References


1. Tu Y, Stolovitzky G, Klein U (2002) Quantitative noise analysis for gene-expression microarray experiments. Proc Natl Acad Sci USA 99: 14031–14036
2. Klebanov L, Yakovlev A (2007) How high is the level of technical noise in microarray data? Biol Direct 2: 9. https://doi.org/10.1186/1745-6150-2-9
3. Bolstad BM, Irizarry RA, Åstrand M *et al* (2003) A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. Bioinformatics 19: 185–193
4. Irizarry RA, Bolstad BM, Collin F *et al* (2003) Summaries of Affymetrix GeneChip probe level data. Nucleic Acids Res 31: e15. https://doi.org/10.1093/nar/gng015
5. Li C, Wong WH (2001) Model-based analysis of oligonucleotide arrays: expression index computation and outlier detection. Proc Natl Acad Sci USA 98: 31–36
6. Hubbell E, Liu WM, Mei R (2002) Robust estimators for expression analysis. Bioinformatics 18: 1585–1592
7. Liu G, Loraine AE, Shigeta R *et al* (2003) NetAffx: Affymetrix probesets and annotations. Nucleic Acids Res 31: 82–86
8. Irizarry RA, Hobbs B, Collin F *et al* (2003) Exploration, normalization, and summaries of high density oligonucleotide array probe level data. Biostatistics 4: 249–264
9. Wu Z (2009) A review of statistical methods for preprocessing oligonucleotide microarrays. Stat Methods Med Res 18: 533–541

10. Cheng L, Lo LY, Tang NLS *et al* (2016) CrossNorm: a novel normalization strategy for microarray data in cancers. Sci Rep 6: 18898. https://doi.org/10.1038/srep18898
11. Astrand M (2003) Contrast normalization of oligonucleotide arrays. J Comput Biol 10: 95–102
12. Workman C, Jensen LJ, Jarmer H *et al* (2002) A new non-linear normalization method for reducing variability in DNA microarray experiments. Genome Biol 3: research0048.1–research0048.16. https://doi.org/10.1186/gb-2002-3-9-research0048
13. Huber W, Von Heydebreck A, Sültmann H *et al* (2002) Variance stabilization applied to microarray data calibration and to the quantification of differential expression. Bioinformatics 18: 96–104
14. Larriba Y, Rueda C, Fernández MA *et al* (2018) A bootstrap based measure robust to the choice of normalization methods for detecting rhythmic features in high dimensional data. Front Genet 9: 24. https://doi.org/10.3389/fgene.2018.00024
15. Slavov N, Airoldi EM, Van Oudenaarden A *et al* (2012) A conserved cell growth cycle can account for the environmental stress responses of divergent eukaryotes. Mol Biol Cell 23: 1986–1997
16. Oliva A, Rosebrock A, Ferrezuelo F *et al* (2005) The cell cycle-regulated genes of Schizosaccharomyces pombe. PLoS Biol 3: 1239–1260
17. Peng X, Karuturi RKM, Miller LD *et al* (2005) Identification of cell cycle-regulated genes in fission yeast. Mol Biol Cell 16: 1026–1042
18. Rustici G, Mata J, Kivinen K *et al* (2004) Periodic gene expression program of the fission yeast cell cycle. Nat Genet 36: 809–817
19. Barragán S, Fernández MA, Rueda C *et al* (2015) Determination of temporal order among the components of an oscillatory system. PLoS One 10: e0124842. https://doi.org/10.1371/journal.pone.0124842
20. Hughes ME, DiTacchio L, Hayes KR (2009) Harmonics of circadian gene transcription in mammals. PLoS Genet 5: e1000442. https://doi.org/10.1371/journal.pgen.1000442
21. Larriba Y, Rueda C, Fernández MA *et al* (2016) Order restricted inference for oscillatory systems for detecting rhythmic genes. Nucleic Acids Res 44: e163. https://doi.org/10.1093/nar/gkw771
22. Levine JD, Funes P, Dowse HB *et al* (2002) Signal analysis of behavioral and molecular cycles. BMC Neurosci 3: 1. https://doi.org/10.1186/1471-2202-3-1
23. Straume M (2004) DNA microarray time series analysis: automated statistical assessment of circadian rhythms in gene expression patterning. Methods Enzymol 383: 149–166
24. Hughes ME, Hogenesch JB, Kornacker K (2010) Jtk-cycle: an efficient nonparametric algorithm for detecting rhythmic components in genome-scale data sets. J Biol Rhythm 25: 372–380
25. Thaben PF, Westermark PO (2014) Detecting rhythms in time series with rain. J Biol Rhythm 29: 391–400
26. Robertson T, Wright FT, Dykstra RL (1988) Order restricted statistical inference. Wiley, New York
27. Fernández MA, Rueda C, Peddada SD (2012) Identification of a core set of signature cell cycle genes whose relative order of time to peak expression is conserved across species. Nucleic Acids Res 40: 2823–2832
28. Peddada SD, Umbach DM, Harris S (2012) Statistical analysis of gene expression studies with ordered experimental conditions. Handbook of statistics. Elsevier, Amsterdam
29. Barragán S, Fernández MA, Rueda C *et al* (2013) isocir: an r package for constrained inference using isotonic regression for circular data, with an application to cell biology. J Stat Sotw 54: i04. https://doi.org/10.18637/jss.v054.i04
30. Suárez MB, Alonso-Nuñez ML, del Rey F *et al* (2015) Regulation of ace2-dependent genes requires components of the PBF complex in Schizosaccharomyces pombe. Cell Cycle 14: 3124–3137
31. Rueda C, Fernández MA, Barragán S *et al* (2016) Circular piecewise regression with applications to cell-cycle data. Biometrics 72: 1266–1274
32. Barragán S, Fernández MA, Rueda C (2017) Circular order aggregation and its application to cell-cycle genes expressions. Bioinformatics 14: 819–829
33. Freudenberg J, Boriss H, Hasenclever D (2004) Comparison of preprocessing procedures for oligo-nucleotide micro-arrays by parametric bootstrap simulation of spike-in experiments. Methods Inform Med 43: 434–438
34. Nykter M, Aho T, Ahdesmäki M *et al* (2006) Simulation of microarray data with realistic characteristics. BMC Bioinformatics 7: 349. https://doi.org/10.1186/1471-2105-7-349
35. Parrish RS, Spencer III HJ, Xu P (2009) Distribution modeling and simulation of gene expression data. Comput Stat Data Anal 53: 1650–1660

36. Dembélé D (2013) A flexible microarray data simulation model. Microarrays 44: 115–130
37. Nagoshi E, Saini C, Bauer C *et al* (2004) Circadian gene expression in individual fibroblasts: Cell-autonomous and self-sustained oscillators pass time to daughter cells. Cell 119: 693–705
38. Baggs JE, Price TS, DiTacchio L *et al* (2009) Network features of the mammalian circadian clock. PLoS Biol 7: 0563–0575
39. Niforou KM, Anagnostopoulos AK, Vougas K *et al* (2008) The proteome profile of the human osteosarcoma u2os cell line. Cancer Genomics Proteomics 5: 63–77
40. Gautier L, Cope L, Bolstad BM *et al* (2004) Affy - analysis of Affymetrix GeneChip data at the probe level. Bioinformatics 20: 307–315
41. Ihaka R, Gentleman R (1996) R: a language for data analysis and graphics. J Comput Graph Stat 5: 299–314
42. Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. J R Stat Soc Ser B Methodol 57: 289–300
43. Efron B, Tibshirani RJ (1994) An introduction to the bootstrap. Chapman & Hall/CRC, Boca Raton
44. Emerson JD, Hoaglin DC (1983) Analysis of two-way tables by medians. Understanding robust and exploratory data analysis. Wiley, New York
45. Pizarro A, Hayer K, Lahens NF *et al* (2013) Circadb: a database of mammalian circadian gene expression profiles. Nucleic Acids Res 41: D1009–D1013. https://doi.org/10.1093/nar/gks1161

# Chapter 10

# HPC Tools to Deal with Microarray Data

Jorge González-Domínguez and Roberto R. Expósito

## Abstract

Parallel and high performance computing is continuously gaining attention in the last years as a means to accelerate several kind of computationally expensive applications. This chapter is a review of different research works and publicly available tools whose target is the acceleration of microarray data analysis, thanks to exploiting high performance computing systems.

**Key words** Microarray data, High performance computing, Parallel computing

## 1 Introduction

High performance computing (HPC) has become increasingly important over the last decades as an essential tool for scientific research. In fact, HPC is currently one of the leading edge disciplines in IT with a wide range of demanding applications in economy, science, and engineering. HPC applications generally involve complex mathematical models and numerical solution techniques that often require a large number of computing resources to cut down their computational complexity.

High-throughput techniques for gene expression analysis have significantly increased its speed during the last years, which has led to a huge amount of microarray data. Scientists and researchers face the challenge of transforming this data into interesting biological information. Therefore, they require scalable bioinformatics tools capable of analyzing large microarray datasets in a reasonable period of time. Meeting HPC and bioinformatics is the solution on several scenarios.

In this chapter we will enumerate a significant number of works that have attempted to accelerate different stages of the microarray analysis pipeline using HPC facilities. Before this enumeration we start the chapter with a brief description of the different kind of parallel architectures that have been exploited for microarray data.

Verónica Bolón-Canedo and Amparo Alonso-Betanzos (eds.), *Microarray Bioinformatics*, Methods in Molecular Biology, vol. 1986, https://doi.org/10.1007/978-1-4939-9442-7_10, 

## 2 Background on HPC Architectures

The main factor driving HPC performance is parallel computing. In its simplest sense, it can be defined as a type of computation in which multiple computing resources are concurrently used to solve a complex computational problem. These computing resources vary widely from a single computer with multiple processors or cores to an arbitrary number of such computers interconnected through a high-speed network. The increasingly computing needs of HPC applications has caused parallel computers to significantly evolve over time. Virtually all standalone computers today are inherently parallel from a hardware perspective, including multiple functional units, processing cores, and hardware threads. In fact, parallelism can be on many levels:

- Micro-architecture level: instruction level parallelism (ILP), single instruction multiple data (SIMD).
- Processor level: multicore processors and manycore accelerators.
- Node level: multiple processors and accelerators per node.
- System level: multiple nodes per system.

Next, this section provides the background of this chapter regarding the HPC architectures and programming models most commonly used to fully exploit the different levels of parallelism available on current systems.

### 2.1 Shared-Memory Systems

Shared-memory systems have in common the ability for all processors or cores to access all the available physical memory as a global address space that they read and write to asynchronously. In this way, multiple processors can operate independently but sharing the same memory resources. So, changes in a memory location carried out by one processor are visible to all other processors in the system. Based upon memory access times, shared-memory systems have historically been classified as:

- Uniform memory access (UMA). Access time to a memory location is independent of which processor makes the request or which memory module contains the data. These systems are also known as symmetric multiprocessor (SMP) machines.
- Non-uniform memory access (NUMA). Not all processors have equal access time to all memories. Each processor still has access to all the available physical memory, but it is directly connected only to a portion of it. To access other parts of the physical memory, it uses a fast interconnection link. So, accessing the local memory is faster than accessing the remote memory.

Representative examples of shared-memory machines are multiprocessor systems, with two or more processors in close

communication, and multicore processors (i.e., a multiprocessor on a single chip). Nowadays, the combination of both approaches is largely being used as the building block for most HPC deployments. Basically, each computing node is usually a NUMA multiprocessor system where multiple processors are installed in different sockets, with each socket providing a UMA multicore processor.

OpenMP [23] is a standardization effort jointly defined by a group of major computer hardware and software vendors, which is the most commonly employed protocol for parallel programming of shared-memory systems. The OpenMP API specification defines a portable and scalable model that provides a simple interface for developing parallel applications in C, C++, and Fortran. Basically, OpenMP is based on the fork-join threading model and proposes a set of compiler directives and library routines. Other lower-level programming models for shared-memory systems are threading libraries such as POSIX threads (pthreads) and the use of inter process communication (IPC) mechanisms (e.g., shared-memory segments) provided by operating systems.

### 2.2 Manycore Accelerators

A recent trend in HPC is the deployment of heterogeneous architectures that combine fine-grain and coarse-grain parallelism using hundreds or thousands of processing cores. These abundant processing cores are generally available in the form of massively parallel hardware devices known as manycore accelerators or co-processors. In HPC, an accelerator is a hardware component whose role is to speed up some aspect of the computing workload. These accelerators can be considered specialized multicore processors designed for a high degree of parallel processing, containing a large number of simpler, independent processing cores. The main advantage of these systems is their good trade-off between computational capabilities and energy requirements. Compared to conventional multicore processors, these devices can offer an order-of-magnitude improvement in performance per watt. Examples of such manycore accelerators include graphics processing units (GPUs), Intel Xeon Phi, and field-programmable gate arrays (FPGAs).

To tackle the platform diversity problem arising from these complex heterogeneous architectures, the open computing language (OpenCL) [106] was proposed in 2008. OpenCL is an open standard maintained by the non-profit technology consortium Khronos Group. Basically, OpenCL is a platform-independent framework to write parallel programs that execute across heterogeneous platforms, proposing a common hardware model for all manycore and multicore platforms. The user programs this "virtual" platform, and the resulting source code is portable on any OpenCL compliant platform. OpenCL views a computing system as consisting of a number of compute devices, which may be general-purpose multicore processors or manycore accelerators attached to a host processor. Moreover, OpenCL defines a

multilevel shared-memory model, featuring four distinct memory spaces (private, local, constant, and global) that are allowed to be collapsed together depending on the memory subsystem of the underlying platform.

#### 2.2.1 GPUs

GPUs were originally designed to accelerate graphics tasks like image rendering, also supporting 3D graphics, especially for video games and graphics APIs such as DirectX and OpenGL. This is basically done through mostly floating-point arithmetic, which is the same stuff HPC developers mainly use supercomputing for. During years, GPUs have evolved from non-programmable hardware pipelines to fully programmable highly parallel manycore architectures, which has enabled their use for non-graphical computationally intensive applications, known as general-purpose computing on GPUs (GPGPU).

To access the GPU computational resources, the algorithms had initially to be expressed in native graphics operations, so efficient applications could be developed but programming was very difficult and time-consuming. For this reason, the tools for the development of GPGPU applications have greatly improved over time. On the one hand, NVIDIA introduced in 2006 the compute unified device architecture (CUDA) [81]. Based on C, CUDA uses language extensions to separate device (i.e., GPU) from host code and data, as well as to launch CUDA kernels. On the other hand, AMD first embraced Brook+ as a high-level programming model for their GPUs, which is based on an extended version of Brook [10], but finally adopted OpenCL.

Basically, programming GPUs is based on offloading and fine-grained parallelism: the host processor offloads the data-parallel kernels as large collections (blocks) of threads on the GPU. From the architectural standpoint, GPUs are inherently SIMD processors that rely on data parallelism with a high number of cores grouped into blocks, and memory organized in a hierarchical way (at thread, block, and global level).

#### 2.2.2 Intel Xeon Phi

Xeon Phi is a series of manycore co-processors based on the many integrated core (MIC) architecture, which has its roots in an earlier GPU design by Intel. The main feature of the MIC architecture is that it provides fully x86-compatible processing cores that can run software that was originally targeted at a standard x86 processor. This means a simpler programming style than that of GPUs, thanks to the use of standard programming languages and APIs such as OpenMP, in addition to platform-independent frameworks such as OpenCL. Basically, MIC is based on simple, low-frequency in-order x86 cores for reduced power consumption together with wide SIMD and multithreading units to deliver massive data and thread parallelism, respectively.

Initially, the first generation of Xeon Phi (i.e., Knights Corner) was only available in the form of PCIe-based cards to be used as manycore accelerators, just like GPUs. However, the second generation (Knights Landing) is available in two forms: as an accelerator or as a host standalone processor.

#### 2.2.3 FPGAs

FPGAs are massively parallel, digital logic arrays designed to be configured by the end user after being manufactured. FPGAs support the notion of reconfigurable computing and offer a high degree of on-chip parallelism, being extremely competitive devices compared to state-of-the-art multicore processors and GPUs while providing lower power consumption.

One of the main hurdles in the utilization of FPGAs as manycore HPC accelerators is the complexity of programming them. The traditional way to configure FPGAs has been through a hardware description language (HDL) such as Verilog or VHDL used by hardware designers. This limitation is being tackled by high-level synthesis (HLS) [72] techniques, which enable designers to program an FPGA using high-level languages (e.g., C, C++). In fact, some vendors are also offering HLS tools for OpenCL to dramatically simplify FPGA programming by enabling users to code their algorithms in a platform-independent parallel framework.

Another major benefit of FPGAs is energy efficiency, reducing power consumption of an HPC system in two ways. First, programmed FPGAs can take on tasks delegated from the host processor to optimize the workload. Second, because the FPGA's role is defined by software, only the needed onboard elements of the FPGA are activated (i.e., the rest of the configurable elements can remain inactive).

### 2.3 Distributed-Memory Systems

The common characteristic of distributed-memory systems is the use of a communication network to connect inter-processor memories. In these systems, processors have their own private memory and there is no mapping of memory addresses across them. Unlike shared-memory systems, there is no concept of a global address space (i.e., changes a processor makes to its local memory have no effect on the memory of other processors). When a processor needs access to data in another processor, it is usually the task of the programmer to explicitly define how and when data is communicated. Distributed-memory systems vary widely from small clusters built using commodity off-the-shelf hardware (e.g., the so-called Beowulf clusters) to large supercomputers installed at national laboratories, big research groups, or large companies.

The message-passing model is by far the most widely used approach to program distributed-memory systems in the HPC community. The message passing interface (MPI) [74] is the de facto standard for message-passing based on the consensus of more than 40 participating organizations, including vendors, researchers,

and users. MPI provides a portable and scalable API to write robust and efficient parallel applications using the message-passing paradigm, where processes exchange data through communications by sending and receiving messages. Currently, there exist several implementations of the MPI interface for C, C++, Fortran, and Java languages.

As mentioned before, the majority of HPC deployments in the world today employ both shared and distributed-memory architectures. These systems provide several multicore nodes connected through a network, where each node is generally equipped with multiple manycore accelerators. To program these systems, hybrid parallel approaches are generally used by combining MPI with a shared-memory thread model (e.g., MPI+OpenMP) or other APIs to exploit the compute capabilities of manycore accelerators (e.g., MPI+CUDA/OpenCL).

### 2.4 Cloud Computing Platforms

Cloud computing [11] is an Internet-based computing model that has gained significant acceptance in many research areas and IT organizations as an elastic, flexible, and variable-cost way to deploy their services using outsourced resources. More specifically, infrastructure as a service (IaaS) is a type of cloud service that dynamically provides on-demand and self-service access to elastic computing resources (e.g., CPU, memory, disk), which are typically billed on a pay-as-you-go basis.

In recent years, cloud computing has generated strong interest in the HPC community due to the effortless access to large number of computing resources. Traditional HPC systems are typically managed and operated by individual organizations in private. However, computing demand is fluctuating, with periods where dedicated resources are either underutilized or overloaded. The advantages of the pay-as-you-go model, elasticity, flexibility, customization, and resource control offered by virtualization technologies make cloud computing an attractive option to meet the needs of HPC users. In this context, resources are no longer hosted by the researchers' computational facilities, but leased from IaaS providers only when needed, which is especially interesting for users who cannot afford their own HPC infrastructure.

The interest of cloud platforms for HPC increases as their availability, computational power, price, and performance improve. Public IaaS providers (e.g., Amazon EC2) are increasingly offering HPC-aimed resources that are intended well suited for HPC workloads and other demanding network-bound applications. These resources include state-of-the-art multicore processors, high-speed networks, enhanced storage performance through solid state drive (SSD) disks, and manycore accelerators (GPUs and FPGAs). Using the powerful abstraction provided by the IaaS model, researchers can set up virtual clusters to exploit

supercomputing-level power without any knowledge of the underlying infrastructure [71].

From the new programming paradigms that have been proposed to deal with large-scale computations on cloud platforms, the MapReduce model developed by Google [24] is the most popular one since its inception. MapReduce executes in parallel several instances of two user-defined functions, *Map* and *Reduce*, over the computing nodes of distributed-memory systems such as clusters and clouds. MapReduce applications are made in batches relying on the distributed Google file system (GFS) [35] to take the input and store the output. Regarding open-source implementations, Hadoop [108] is the most widespread MapReduce framework for large-scale batch processing, providing the Hadoop distributed file system (HDFS) [104] as GFS counterpart. However, MapReduce is not well suited for iterative algorithms because there is no efficient way of reusing data or computation from previous iterations. Emerging frameworks, such as Spark [116] and Flink [14], are designed from the very beginning to efficiently support iterative workloads, providing a programming API based on distributed in-memory data structures (e.g., Spark RDDs [115]). Moreover, these frameworks are suitable for streaming computations, even with real-time or near real-time constraints.

## 3 Acceleration Microarray Data Processing

### 3.1 Preprocessing Steps

In recent years, biologists have generated massive amounts of microarray data using modern platforms. However, this extreme speed in sampling microarray data usually comes with a prize: some data might have low quality or even sampling errors. The first step in microarray analyses consists in assessing the quality of the microarray data and preprocessing the data to obtain gene expressions. Tools that exploit HPC facilities are necessary in order to perform the quality assessment and preprocessing of very large microarray datasets.

Probably the most used parallel tool for microarray data preprocessing is `affyPara` [101]. It extends the sequential `affy` package [52], is written with R and MPI, and is publicly available [89]. `affyPara` involves three steps: background correction, normalization, and summarization. The background correction step attempts to remove the background noise from the raw intensity. Therefore background correction is essential, since part of the measured probe intensities is due to non-specific hybridization and the noise in the optical detection system. Normalization aims to correct the variation of gene expression in the same array due to experimental bias. Summarization combines the multiple probe intensities for each probeset to produce expression values.

A counterpart of `affyPara` is presented in [44]. It also employs a master–slave MPI approach for the normalization and summarization steps. According to the authors it presents three advantages over `affyPara`: (1) the possibility to apply more summarization schemes such as Plier; (2) the mechanism to easily extend the algorithm to newer types of SNP arrays; and (3) it does not require the installation of the bioconductor platform. However, it is not available to download from a public repository.

The two previous MPI approaches need the microarray data loaded into memory for preprocessing, which limits the size of the microarray datasets they can handle. An integrated microarray quality assessment and preprocessing tool parallelized for cloud and GPU systems is capable to handle background correction, normalization, and summarization for larger datasets [82]. It employs Hadoop for the data-intensive part while CUDA for the compute-intensive tasks. A different approach to remove noise of microarray data on the cloud is presented in [45]. Concretely, it uses Hadoop and wavelet-based thresholds to remove noise from microarrays while retaining gene significance within the microarray dataset.

Image processing techniques are also very employed to complete the preprocessing steps and evaluate the gene expressions. Their high complexity makes them suitable for parallelization. For instance, an automated FPGA-based system for microarray image processing is presented in [7]. It includes image enhancement (improve image quality and enhance weakly expressed spots), image addressing or gridding (creation of a grid that matches the spots in the image), and image segmentation (identify those pixels that belong to the microarray spot and those pixels that represent background information). It was satisfactorily tested on a Xilinx xc5vlx110t FPGA found on the Virtex5 platform. FPGAs were also used in [60] to parallelize an edge detection method that can be employed as base for image segmentation of DNA microarray data.

An interesting available tool [109] for image processing of microarray data is `MIGS-GPU` [58], which provides a CUDA implementation of the gridding and segmentation steps for NVIDIA GPUs. They are implemented with a genetic and a grow-cut algorithm (region growing algorithm that can segment a greyscale or multichannel image into regions belonging to multiple classes), respectively. The authors have also presented a OpenMP implementation of their image segmentation approach for multicore CPU systems [57], where threads apply the segmentation to different quadrilaterals of the image provided by the gridding process.

Finally, `PIMA(GE)` [33] is a complete library for image processing that is parallelized with MPI and CUDA to be used on multicore clusters, GPUs, and even on modern heterogeneous clusters and supercomputers where the nodes have available several CPUs

and GPUs. Among different scenarios it has been tested on images obtained with tissue microarray technology, obtaining high scalability.

### 3.2 Construction of Gene Networks

Gene co-expression networks are used to illustrate the complex interactions that can be present among multiple genes. The nodes and edges represent genes and interesting correlations, respectively. There exist several applications that construct these networks receiving as input the microarray expression values for each gene and each sample.

One of the most popular tools to construct gene networks is `ARACNE` [70]. It uses an information theoretic framework based on the data processing inequality theorem to infer direct regulatory relationships between transcriptional regulator proteins and target genes. However, its MATLAB/C++ implementation is quite slow and it is limited to small datasets. A publicly available [5] multi-threaded Java counterpart `ARACNe-AP` [63] that can be executed in parallel on multicore shared-memory systems has been developed. Networks inferred by `ARACNe-AP` are virtually identical to those inferred by the original tool but in significant lower runtime.

There exist different algorithms to construct gene networks, some of them more suitable for parallelization. For instance, `FastGCN` [66] is an available tool [31] that uses OpenMP and CUDA to parallelize its four steps, so that they can be executed either on multicore platforms or GPUs: (1) preprocessing of the input data with genetic information entropy; (2) computation of the Pearson correlation coefficient for each gene pair; (3) transformation of the coefficients to a normal distribution; and (4) identification of modules. Another approach valid for both multicore and GPU systems was presented in [118].

The computational capabilities of GPUs are also exploited by `CUDA-MI` [21] to accelerate the estimation of the pairwise mutual information from gene microarray data [103], which is a common step of many approaches to construct gene networks. `CUDA-GRN` [4] is an efficient GPU version of the widely used `GENIE3` tool [53], which employs random forests to extract gene networks from microarray data. Other examples of GPU-based works are based on random matrix theory [51] or mean conditional entropy [9].

GPUs are not the only type of accelerators that have been used for gene network construction. For instance, an approach based on the parallel calculation of mutual information on Intel Xeon Phi co-processors was presented in [75]. There also exist alternatives for FPGAs focused on Bayesian learning [6, 93] or boolean graphs [32].

The construction of gene networks is so computationally expensive that also larger infrastructures have been exploited to accelerate it. The first approximation using MPI was presented in [98], with a promising experimental evaluation on an old Beowulf

cluster. More modern MPI approaches that produce high-quality networks from large microarray datasets on a reasonable time are described in [107, 121]. Nevertheless, up to our knowledge, `MPI-GeneNet` [36] is the only available tool [79] to accelerate gene network construction with the message-passing paradigm. It efficiently exploits multicore clusters with a two-level MPI+OpenMP parallel implementation of its two steps: (1) Pearson correlation and (2) the random matrix theory threshold. Even an MPI+CUDA implementation for heterogeneous clusters exists [13].

Finally, it is interesting to mention similar approaches also developed for high performance facilities such as clusters or clouds with the MapReduce paradigm. The examples include a Spark implementation of a method based on mutual information [18] and a Hadoop program for time-series microarray data [2] (parallelization of `TimeDelay-ARACNE` [122]). It is worthy to include in this review a work [112] that evaluates the impact of using the RHIPE MapReduce framework (written on top of Hadoop) to parallelize four different correlations methods that are usually an important step for the construction of gene networks.

### 3.3 Statistical Analysis

After a gene expression experiment has been conducted and basic preprocessing has been performed, the general analysis procedure for univariate association testing involves comparing samples with a combination of criteria for expression level and statistical significance. For a typical case-control study, probably the most common approach consists in the application of one or several statistical tests to compare expression levels for each gene or SNP of interest in control individuals to those that present the disease. These tests are not usually too expensive in terms of computational time, so they do not require extreme parallel solutions. Nevertheless, a publicly available [26] multithreaded library called `EDGE` efficiently works on small multicore shared-memory systems either for static or time-course data [65]. There even exist cloud-based approaches for tests such as ANOVA [95] or Fisher's [3, 25].

These kind of tests have been extensively used for analysis due to its good trade-off between simplicity and quality of results. Nevertheless, alternatives based on machine learning techniques such as feature selection or classification can increase the accuracy of the analyses at expense of longer runtimes. These approaches can benefit from high performance computing. For instance, the available CUDA tools `CFMDS` [15] and `FastMRMR` [29] perform feature selection on NVIDIA GPUs and have been tested over microarrays. The first tool implements the multidimensional scaling method [90] while the second provides a greedy optimization of the popular mRMR (minimum redundancy maximum relevance) feature selection approach [94]. Examples of classification techniques parallelized for GPUs are presented in [8] and [113], while an alternative for FPGAs is presented in [50]. Some examples for distributed-

memory systems are the MPI parallelization of the multicategory support vector machine (SVM) classification method [117], and the random forest classifier implemented with parallel R [76, 77]. Nevertheless, MapReduce paradigm is the most popular when gathering machine learning and microarray data. Approaches focus on different classification methods such as K-nearest neighbor [54, 62] or proximal SVM [61], and some of them also include parallel feature selection algorithms [96].

The use of computationally expensive data mining techniques such as clustering or biclustering can also be useful to associate genes or SNPs to the presence or absence of certain diseases. Parallel works related to these techniques compress all possible HPC architectures. Starting by Intel shared-memory platforms, a parallel dictionary learning can accelerate the biclustering of microarray gene expression data on multicore CPU platforms [64], while the partition around medoids algorithm (a variation of K-Means) can benefit from the Xeon Phi co-processor [97]. Applications for specific hardware include several FPGA-based implementations of K-Means [12, 49], as well as GPU the publicly available library `CAMPAIGN` [20], that provides GPU implementations of several clustering methods [59]. Three open-source tools (DBSCAN [86], PINK [87], and OPTICS [88]) are the most representative examples of clustering using the message-passing paradigm. They implement density-based clustering [91], hierarchical clustering [47], and a mix of both methods [92], respectively.

Finally, some works even deal with different architectures and compare their suitability for these data mining algorithms. For instance, the performance of both multicore CPUs and GPUs biclustering or microarrays is analyzed in [67] and [84] (this last work also includes FPGAs). The hybrid MPI+CUDA implementation of `NMF-mGPU` [73] allows this publicly available tool [83] to exploit heterogeneous clusters where each node contains one or several GPUs to complete the biclustering of microarrays through the non-negative matrix factorization method. A hybrid parallel implementation of the K-Means clustering technique that exploits both CPU cores and FPGAs on a single node has been presented in [1]. There also exists an implementation of this algorithm for clusters of FPGAs, using Hadoop to distribute the workload among accelerators [19].

### 3.4 Epistasis Detection

The main drawback of the univariate tests presented in the previous section is that they are not powerful enough to explain most of genetic influence on diseases. In these cases the detection of joint genetic effects of two or more SNPs or genes (epistasis) needs to be considered [78]. The most common approach consists in checking the influence of SNP-pairs (second order interactions). Algorithms that address this kind of problem are expensive due to their quadratic complexity with the number of SNPs. Many parallel tools

have been developed, and even two reviews have been published [16, 111], the second one only focused on GPUs.

GPUs are undoubtedly the most popular parallel platform to accelerate second level epistasis. As examples of publicly available tools we should mention `MDRGPU` [80] (a CUDA implementation of the powerful model-free multifactor dimensionality reduction method [42]), `SHEsisEpi` [100] (using Z-test to determine the SNP-pairs with influence on the disease [48]), `EpiGPU` [27] (OpenCL implementation that can be executed on GPUs of different vendors [46]), and `GBOOST` [34] (with boolean data representation to accelerate calculations [114]). There also exist research works that compare the performance of GPUs implementations with other parallel architectures such as multicore CPUs [17, 38], FPGAs [39], or Intel Xeon Phi co-processors [22, 40, 105].

Nevertheless, larger HPC facilities have also been exploited to accelerate pairwise epistasis detection with message-passing-based tools such as `EPISNPmpi` [28] and `FastEpistasis` [30] (both of them implemented with MPI for quantitative analysis [68, 102]), or `ParallABEL` [85] (implemented with Rmpi [99]). Cloud-based counterparts include tools based on ant colony [120] or chi-square tests [119].

Although less common, it is even possible to analyze the influence of larger groups of SNPs or genes. Remark `GPU3SNP` [41] and `TST` [110] as tools that are able to exploit GPUs to find third-level epistasis (interactions of SNP-triplets) [37, 69]. A similar alternative for FPGAs was presented in [56]. Finally, parallel implementations to find interactions with a level even higher than three were developed for GPUs [55] and cloud systems [43].

## References


1. Abdelrahman TS (2016) Accelerating K-means clustering on a tightly-coupled processor-FPGA heterogeneous system. In: 2016 IEEE international conference on application-specific systems, architectures and processors (ASAP), pp 176–181
2. Abduallah Y, Turki T, Byron K, Du Z, Cervantes-Cervantes M, Wang JTL (2017) MapReduce algorithms for inferring gene regulatory networks from time-series microarray data using an information-theoretic approach. BioMed Res Int. https://doi.org/10.1155/2017/6261802
3. Agapito G, Cannataro M, Guzzi PH, Marozzo F, Talia D, Trunfio P (2013) Cloud4SNP: distributed analysis of SNP microarray data on the cloud. In: 2013 International conference on bioinformatics, computational biology and biomedical informatics (BCB), p 468
4. Alborzi SZ, Maduranga DAK, Fan R, Rajapakse JC, Zheng J (2014) CUDAGRN: parallel speedup of inferring large gene regulatory networks from expression data using random forest. In: 2014 IAPR international conference on pattern recognition in bioinformatics (PRIB), pp 85–97
5. ARACNe-AP: network reverse engineering through AP inference of mutual information (2018). https://sourceforge.net/projects/aracne-ap/. Last accessed March 2018
6. Asadi NB, Fletcher CW, Gibeling G, Glass EN, Sachs K, Burke D, Zhou Z, Wawrzynek J, Wong WH, Nolan GP (2010) Paralearn: a massively parallel, scalable system for learning interaction networks on FPGAs. In: 2010 ACM international conference on supercomputing (SC), pp 83–94
7. Belean B, Borda M, Le Gal B, Terebes R (2012) FPGA based system for automatic

cDNA microarray image processing. Comput Med Imaging Graph 36(5):419–429
8. Benso A, Di Carlo S, Politano G, Savino A (2010) GPU acceleration for statistical gene classification. In: 2010 IEEE international conference on automation quality and testing robotics (AQTR), vol 2, pp 1–6
9. Borelli FF, de Camargo RY, Martins DC, Rozante LCS (2013) Gene regulatory networks inference using a multi-GPU exhaustive search algorithm. BMC Bioinformatics 14(18):S5
10. Buck I, Foley T, Horn D, Sugerman J, Fatahalian K, Houston M *et al* (2004) Brook for GPUs: stream computing on graphics hardware. ACM Trans Graph 23(3):777–786
11. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Futur Gener Comput Syst 25(6):599–616
12. Canilho J, Véstias M, Neto H (2016) Multicore for K-means clustering on FPGA. In: 2016 International conference on field programmable logic and applications (FPL), pp 1–4
13. Carastan-Santos D, de Camargo RY, Martins DC, Song SW, Rozante LCS, Borelli FF (2015) A multi-GPU hitting set algorithm for GRNs inference. In: 2015 IEEE/ACM international symposium on cluster, cloud and grid computing (CCGrid), pp 313–322
14. Carbone P, Katsifodimos A, Ewen S, Markl V, Haridi S, Tzoumas K (2015) Apache Flink: stream and batch processing in a single engine. Bull IEEE Comput Soc Tech Comm Data Eng 38(4):28–38
15. CFMDS (CUDA-based fast multidimensional scaling) (2018). http://ml.ssu.ac.kr/CFMDS/CFMDS.html. Last accessed March 2018
16. Chen GK, Guo Y (2013) Discovering epistasis in large scale genetic association studies by exploiting graphics cards. Front Genet 4:266
17. Chikkagoudar S, Wang K, Li M (2011) GENIE: a software package for gene-gene interaction analysis in genetic association studies using multiple GPU or CPU cores. BMC Res Notes 4(1):158
18. Chockalingam SP, Aluru M, Aluru S (2015) Information theory based genome-scale gene networks construction using mapreduce. In: 2015 IEEE international conference on high performance computing (HiPC), pp 464–473
19. Choi Y-M, So HK-H (2014) Map-reduce processing of K-means algorithm with FPGA-accelerated computer cluster. In: 2014 IEEE international conference on application-specific systems, architectures and processors (ASAP), pp 9–16
20. Clustering algorithms for massively parallel architectures including GPU nodes (2018). https://simtk.org/projects/campaign. Last accessed March 2018
21. CUDA-MI (2018). https://sites.google.com/site/liuweiguohome/cuda-mi. Last accessed March 2018
22. Curk T, Rot G, Zupan B (2011) SNPsyn: detection and exploration of SNP–SNP interactions. Nucleic Acids Res 39(suppl_2): W444–W449
23. Dagum L, Menon R (1998) OpenMP: an industry standard API for shared-memory programming. IEEE Comput Sci Eng 5 (1):46–55
24. Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. Commun ACM 51(1):107–113
25. Dudley JT, Pouliot Y, Chen R, Morgan AA, Butte AJ (2010) Translational bioinformatics in the cloud: an affordable alternative. Genome Med 2(8):51
26. Edge: R package for identifying differentially expressed genes from genome-wide gene expression profiling studies (2018). https://github.com/StoreyLab/edge. Last accessed March 2018
27. epiGPU v2.0 (2018). https://github.com/explodecomputer/epiGPU. Last accessed March 2018
28. EPISNPmpi Homepage (2018). https://animalgene.umn.edu/episnpmpi. Last accessed March 2018
29. fast-mRMR (2018). https://github.com/sramirez/fast-mRMR. Last accessed March 2018
30. FastEpistasis Homepage (2018). http://www.vital-it.ch/software/FastEpistasis. Last accessed March 2018
31. FastGCN for gene co-expression network (2018). https://github.com/DrLiang/FastGCN. Last accessed March 2018
32. Ferreira R, Vendramini JCG (2010) FPGA-accelerated attractor computation of scale free gene regulatory networks. In: 2010 international conference on field programmable logic and applications (FPL), pp 550–555
33. Galizia A, D'Agostino D, Clematis A (2015) An MPI–CUDA library for image processing on HPC architectures. J Comput Appl Math 273:414–427

34. GBOOST Homepage (2018). http://bioinformatics.ust.hk/BOOST.html#GBOOST. Last accessed March 2018
35. Ghemawat S, Gobioff H, Leung S-T (2003) The Google file system. SIGOPS Oper Syst Rev 37(5):29–43
36. González-Domínguez J, Martín MJ (2017) MPIGeneNet: parallel calculation of gene co-expression networks on multicore clusters. IEEE/ACM Trans Comput Biol Bioinform 15(5):1732–1737
37. González-Domínguez J, Schmidt B (2015) GPU-accelerated exhaustive search for third-order epistatic interactions in case–control studies. J Comput Sci 8:93–100
38. González-Domínguez J, Schmidt B, Kässens JC, Wienbrandt L (2014) Hybrid CPU/GPU acceleration of detection of 2-SNP epistatic interactions in GWAS. In: 2014 European conference on parallel processing (Euro-Par), pp 680–691
39. Gonzalez-Dominguez J, Wienbrandt L, Kassens JC, Ellinghaus D, Schimmler M, Schmidt B (2015) Parallelizing epistasis detection in GWAS on FPGA and GPU-accelerated computing systems. IEEE/ACM Trans Comput Biol Bioinform 12(5):982–994
40. González-Domínguez J, Ramos S, Touriño J, Schmidt B (2016) Parallel pairwise epistasis detection on heterogeneous computing architectures. IEEE Trans Parallel Distrib Syst 27 (8):2329–2340
41. GPU3SNP: exhaustive search for third order epistatic interactions using CUDA (2018). https://sourceforge.net/projects/gpu3snp/. Last accessed March 2018
42. Greene CS, Sinnott-Armstrong NA, Himmelstein DS, Park PJ, Moore JH, Harris BT (2010) Multifactor dimensionality reduction for graphics processing units enables genome-wide testing of epistasis in sporadic ALS. Bioinformatics 26(5):694–695
43. Guo X, Meng Y, Yu N, Pan Y (2014) Cloud computing for detecting high-order genome-wide epistatic interaction via dynamic clustering. BMC Bioinformatics 15(1):102
44. Guzzi PH, Cannataro M (2010) Parallel Pre-processing of Affymetrix microarray data. In: 2010 European conference on parallel processing, Euro-Par, pp 225–232
45. Harvey BS, Ji S-Y (2017) Cloud-scale genomic signals processing for robust large-scale cancer genomic microarray data analysis. IEEE J Biomed Health Inform 21 (1):238–245
46. Hemani G, Theocharidis A, Wei W, Haley C (2011) EpiGPU: exhaustive pairwise epistasis scans parallelized on consumer level graphics cards. Bioinformatics 27(11):1462–1465
47. Hendrix W, Palsetia D, Patwary MdMA, Agrawal A, Liao W-K, Choudhary A (2013) A scalable algorithm for single-linkage hierarchical clustering on distributed-memory architectures. In: 2013 IEEE symposium on large-scale data analysis and visualization (LDAV), pp 7–13
48. Hu X, Liu Q, Zhang Z, Li Z, Wang S, He L, Shi Y (2010) SHEsisEpi, a GPU-enhanced genome-wide SNP-SNP interaction scanning algorithm, efficiently reveals the risk genetic epistasis in bipolar disorder. Cell Res 20 (7):854
49. Hussain HM, Benkrid K, Seker H, Erdogan AT (2011) FPGA implementation of K-means algorithm for bioinformatics application: an accelerated approach to clustering microarray data. In: 2011 NASA/ESA conference on adaptive hardware and systems (AHS), pp 248–255
50. Hussain HM, Benkrid K, Seker H (2013) Reconfiguration-based implementation of SVM classifier on FPGA for classifying microarray data. In: 2013 Annual international conference of the IEEE engineering in medicine and biology society (EMBC), pp 3058–3061
51. Ingram J, Zhu M (2011) GPU accelerated microarray data analysis using random matrix theory. In: 2011 IEEE international conference on high performance computing and communications (HPCC), pp 839–844
52. Irizarry RA, Gautier L, Cope LM *et al* (2003) An R package for analyses of Affymetrix oligonucleotide arrays. In: The analysis of gene expression data. Springer, New York, pp 102–119
53. Irrthum A, Wehenkel L, Geurts P *et al* (2010) Inferring regulatory networks from expression data using tree-based methods. PLoS One 5(9):e12776
54. Islam AKMT, Jeong B-S, Bari ATMG, Lim C-G, Jeon S-H (2015) MapReduce based parallel gene selection method. Appl Intell 42 (2):147–156
55. Jünger D, Hundt C, Domínguez JG, Schmidt B (2017) Speed and accuracy improvement of higher-order epistasis detection on CUDA-enabled GPUs. Clust Comput 20 (3):1899–1908
56. Kässens JC, Wienbrandt L, González-Domínguez J, Schmidt B, Schimmler M (2015) High-speed exhaustive 3-locus interaction epistasis analysis on FPGAs. J Comput Sci 9:131–136

57. Katsigiannis S, Zacharia E, Maroulis D (2015) Grow-cut based automatic cDNA microarray image segmentation. IEEE Trans Nanobioscience 14(1):138–145
58. Katsigiannis S, Zacharia E, Maroulis D (2017) MIGS-GPU: microarray image gridding and segmentation on the GPU. IEEE J Biomed Health Inform 21(3):867–874
59. Kohlhoff KJ, Sosnick MH, Hsu WT, Pande VS, Altman RB (2011) CAMPAIGN: an open-source library of GPU-accelerated data clustering algorithms. Bioinformatics 27 (16):2321–2322
60. Kornaros G (2010) A soft multi-core architecture for edge detection and data analysis of microarray images. J Syst Archit 56(1):48–62
61. Kumar M, Rath SK (2015) Classification of microarray using mapreduce based proximal support vector machine classifier. Knowl-Based Syst 89:584–602
62. Kumar M, Rath NK, Rath SK (2016) Analysis of microarray leukemia data using an efficient mapreduce-based K-nearest-neighbor classifier. J Biomed Inform 60:395–409
63. Lachmann A, Giorgi FM, Lopez G, Califano A (2016) ARACNe-AP: gene network reverse engineering through adaptive partitioning inference of mutual information. Bioinformatics 32(14):2233–2235
64. Laide S, McAllister J (2017) Multicore distributed dictionary learning: a microarray gene expression biclustering case study. In: 2017 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 1168–1172
65. Leek JT, Monsen E, Dabney AR, Storey JD (2005) EDGE: extraction and analysis of differential gene expression. Bioinformatics 22 (4):507–508
66. Liang M, Zhang F, Jin G, Zhu J (2015) FastGCN: a GPU accelerated tool for fast gene co-expression networks. PLoS One 10 (1):e0116776
67. Liu B, Yu CW, Wang DZ, Cheung RCC, Yan H (2014) Design exploration of geometric biclustering for microarray data analysis in data mining. IEEE Trans Parallel Distrib Syst 25(10):2540–2550
68. Ma L, Runesha HB, Dvorkin D, Garbe JR, Da Y (2008) Parallel and serial computing tools for testing single-locus and epistatic SNP effects of quantitative traits in genome-wide association studies. BMC Bioinformatics 9 (1):315
69. Magis AT, Earls JC, Ko Y-H, Eddy JA, Price ND (2011) Graphics processing unit implementations of relative expression analysis algorithms enable dramatic computational speedup. Bioinformatics 27(6):872–873
70. Margolin AA, Nemenman I, Basso K, Wiggins C, Stolovitzky G, Favera RD, Califano A (2006) ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. BMC Bioinformatics 7(1):S7
71. Mauch V, Kunze M, Hillenbrand M (2013) High performance cloud computing. Futur Gener Comput Syst 29(6):1408–1416
72. Meeus W, Van Beeck K, Goedemé T, Meel J, Stroobandt D (2012) An overview of today's high-level synthesis tools. Des Autom Embed Syst 16(3):31–51
73. Mejía-Roa E, Tabas-Madrid D, Setoain J, García C, Tirado F, Pascual-Montano A (2015) NMF-mGPU: non-negative matrix factorization on multi-GPU systems. BMC Bioinformatics 16(1):43
74. Message Passing Interface Forum (1994) MPI: a Message Passing Interface standard
75. Misra S, Pamnany K, Aluru S (2015) Parallel mutual information based construction of genome-scale networks on the Intel® Xeon Phi™Coprocessor. IEEE/ACM Trans Comput Biol Bioinform 12(5):1008–1020
76. Mitchell L, Sloan TM, Mewissen M, Ghazal P, Forster T, Piotrowski M, Trew AS (2011) A parallel random forest classifier for R. In: 2011 International workshop on emerging computational methods for the life sciences (ECMLS), pp 1–6
77. Mitchell L, Sloan TM, Mewissen M, Ghazal P, Forster T, Piotrowski M, Trew A (2014) Parallel classification and feature selection in microarray data using SPRINT. Concurr Comput Pract Experience 26(4):854–865
78. Moore JH, Asselbergs FW, Williams SM (2010) Bioinformatics challenges for genome-wide association studies. Bioinformatics 26(4):445–455
79. MPIGeneNet: parallel tool to construct gene co-expression networks (2018). https://sourceforge.net/projects/mpigenenet/. Last accessed March 2018
80. Multifactor dimensionality reduction (2018). https://sourceforge.net/projects/mdr/. Last accessed March 2018
81. Nickolls J, Buck I, Garland M, Skadron K (2008) Scalable parallel programming with CUDA. In: 35th International conference on computer graphics and interactive techniques (SIGGRAPH'08), pp 16:1–16:14
82. Niu S, Yang G, Sarma N, Xuan P, Smith MC, Srimani P, Luo F (2014) Combining Hadoop and GPU to preprocess large Affymetrix

microarray data. In: 2014 IEEE international conference on big data (Big Data), pp 692–700

83. NMF-mGPU: non-negative matrix factorization on multi-GPU systems (2018). http://bioinfo-cnb.github.io/bionmf-gpu/. Last accessed March 2018
84. Orzechowski P, Boryczko K (2015) Rough assessment of GPU capabilities for parallel PCC-based biclustering method applied to microarray data sets. Bio-Algorithms Med--Syst 11(4):243–248
85. ParallABEL: an R library for generalized parallelization of genome-wide association studies (2018). http://www.sc.psu.ac.th/units/genome/CGBR/ParallABEL/. Last accessed March 2018
86. Parallel DBSCAN Code Download (2018). http://cucis.ece.northwestern.edu/projects/Clustering/download_code_dbscan.html. Last accessed March 2018
87. Parallel hierarchical clustering code download (2018). http://cucis.ece.northwestern.edu/projects/Clustering/download_code_pink.html. Last accessed March 2018
88. Parallel OPTICS code download (2018). http://cucis.ece.northwestern.edu/projects/Clustering/download_code_optics.html. Last accessed March 2018
89. Parallelized preprocessing methods for affymetrix oligonucleotide array (2018). https://bioconductor.org/packages/release/bioc/html/affyPara.html. Last accessed March 2018
90. Park S, Shin S-Y, Hwang K-B (2012) CFMDS: CUDA-based fast multidimensional scaling for genome-scale data. BMC Bioinformatics 13(17):S23
91. Patwary MA, Palsetia D, Agrawal A, Liao W-K, Manne F, Choudhary A (2012) A new scalable parallel DBSCAN algorithm using the disjoint-set data structure. In: 2012 International conference on high performance computing, networking, storage and analysis (SC), p 62
92. Patwary MA, Palsetia D, Agrawal A, Liao W-K, Manne F, Choudhary A (2013) Scalable parallel OPTICS data clustering using graph algorithmic techniques. In: 2013 International conference for high performance computing, networking, storage and analysis (SC), pp 1–12
93. Pournara I, Bouganis C-S, Constantinides GA (2005) FPGA-accelerated Bayesian learning for reconstruction of gene regulatory networks. In: 2005 International conference on field programmable logic and applications (FPL), pp 323–328
94. Ramírez-Gallego S, Lastra I, Martínez-Rego D, Bolón-Canedo V, Benítez JM, Herrera F, Alonso-Betanzos A (2017) Fast-mRMR: fast minimum redundancy maximum relevance algorithm for high-dimensional big data. Int J Intell Syst 32(2):134–152
95. Ray RB, Kumar M, Rath SK (2016) Fast computing of microarray data using resilient distributed dataset of Apache Spark. In: Recent advances in information and communication technology 2016. Springer, Cham, pp 171–182
96. Ray RB, Kumar M, Tirkey A, Rath SK (2016) Scalable information gain variant on spark cluster for rapid quantification of microarray. Procedia Comput Sci 93:292–298
97. Rechkalov T, Zymbler M (2015) Accelerating medoids-based clustering with the Intel many integrated core architecture. In: 2015 International conference on application of information and communication technologies (AICT), pp 413–417
98. Ruchkys DP, Song SW (2003) A parallel solution to infer genetic network architectures in gene expression analysis. Int J High Perform Comput Appl 17(2):163–172
99. Sangket U, Mahasirimongkol S, Chantratita W, Tandayya P, Aulchenko YS (2010) ParallABEL: an R Library for generalized parallelization of genome-wide association studies. BMC Bioinformatics 11 (1):217
100. SHEsis main (2018). http://analysis.bio-x.cn/. Last accessed March 2018
101. Schmidberger M, Vicedo E, Mansmann U (2009) affypara—a bioconductor package for parallelized preprocessing algorithms of Affymetrix microarray data. Bioinf Biol Insights 3:83
102. Schüpbach T, Xenarios I, Bergmann S, Kapur K (2010) FastEpistasis: a high performance computing solution for quantitative trait epistasis. Bioinformatics 26(11):1468–1469
103. Shi H, Schmidt B, Liu W, Müller-Wittig W (2011) Parallel mutual information estimation for inferring gene regulatory networks on GPUs. BMC Res Notes 4(1):189
104. Shvachko K, Kuang H, Radia S, Chansler R (2010) The Hadoop distributed file system. In: IEEE 26th symposium on mass storage systems and technologies (MSST'2010), pp 1–10
105. Sluga D, Curk T, Zupan B, Lotric U (2014) Heterogeneous computing architecture for

fast detection of SNP-SNP interactions. BMC Bioinformatics 15(1):216

106. Stone JE, Gohara D, Shi G (2010) OpenCL: a parallel programming standard for heterogeneous computing systems. Comput Sci Eng 12(3):66–73
107. Tamada Y, Imoto S, Araki H, Nagasaki M, Print C, Charnock-Jones DS, Miyano S (2011) Estimating genome-wide gene networks using nonparametric Bayesian network models on massively parallel computers. IEEE/ACM Trans Comput Biol Bioinform 8(3):683–697
108. The Apache Software Foundation (2006). Apache Hadoop
109. The real-time systems and Image Analysis Lab (2018). http://rtsimage.di.uoa.gr/. Last accessed March 2018
110. Top-scoring pair and top-scoring triple on the graphics processing unit (2018). https://www.igb.illinois.edu/labs/price/downloads/. Last accessed March 2018
111. Upton A, Trelles O, Cornejo-García JA, Perkins JR (2015) High-performance computing to detect epistasis in genome scale data sets. Brief Bioinform 17(3):368–379
112. Wang S, Pandis I, Johnson D, Emam I, Guitton F, Oehmichen A, Guo Y (2014) Optimising parallel R correlation matrix calculations on gene expression data using mapreduce. BMC Bioinformatics 15(1):351
113. Wu H-C, Wei X-G, Chan S-C (2017) Novel consensus gene selection criteria for distributed GPU partial least squares-based gene microarray analysis in diffused large B cell lymphoma (DLBCL) and related findings. IEEE/ACM Trans Comput Biol Bioinform 15(6):2039–2052
114. Yung LS, Yang C, Wan X, Yu W (2011) GBOOST: a GPU-based tool for detecting gene–gene interactions in genome–wide case control studies. Bioinformatics 27 (9):1309–1310
115. Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M *et al* (2012) Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: 9th USENIX symposium on networked systems design and implementation (NSDI'12), pp 15–28
116. Zaharia M, Xin RS, Wendell P, Das T, Armbrust M, Dave A *et al* (2016) Apache Spark: a unified engine for Big Data processing. Commun ACM 59(11):56–65
117. Zhang C, Li P, Rajendran A, Deng Y, Chen D (2006) Parallelization of multicategory support vector machines (PMC-SVM) for classifying microarray data. BMC Bioinformatics 7 (4):S15
118. Zheng M, Zhuo M, Zhang S, Liu G (2017) Inferring genome-wide gene regulatory networks with GPU or CPU parallel algorithm. In: 2017 International conference on computer network, electronic and automation (ICCNEA), pp 54–58
119. Zhou Z, Liu G, Su L, Yan L, Han L (2013) CChi: an efficient cloud epistasis test model in human genome wide association studies. In: 2013 International conference on biomedical engineering and informatics (BMEI), pp 787–791
120. Zhou Z, Liu G, Su L (2016) A new approach to detect epistasis utilizing parallel implementation of ant colony optimization by mapreduce framework. Int J Comput Math 93 (3):511–523
121. Zola J, Aluru M, Sarje A, Aluru S (2010) Parallel information-theory-based construction of genome-wide gene regulatory networks. IEEE Trans Parallel Distrib Syst 21 (12):1721–1733
122. Zoppoli P, Morganella S, Ceccarelli M (2010) TimeDelay-ARACNE: reverse engineering of gene networks from time-course data by an information theoretic approach. BMC Bioinformatics 11(1):154

# Chapter 11

# ROC Curves for the Statistical Analysis of Microarray Data

**Ricardo Cao and Ignacio López-de-Ullibarri**

## Abstract

A receiver operating characteristic (ROC) curve is a graphical plot that illustrates the diagnostic ability of a binary classifier as a function of its discrimination threshold. This chapter is an overview on the use of ROC curves for microarray data. The notion of ROC curve and its motivation is introduced in Subheading 1. Relevant scientific contributions concerning the use of ROC curves for microarray data are briefly reviewed in Subheading 2. The special case with covariates is considered in Subheading 3. Two relevant aspects are reviewed in this section: the use of LASSO techniques for selecting and combining relevant markers and how to correct for multiple testing when a large number of markers are available. Finally, some conclusions are included.

**Key words** AUC, FDR, FWER, LASSO, Microarray, Multiple testing, pAUC, ROC curve

## 1 Motivation and Background

A diagnostic test is a procedure to identify an individual's propensity to disease or illness. For example, in order to diagnose a herniated disc, physicians may employ magnetic resonance imaging (MRI), computer assisted tomography (CAT scan), and/or electromyography (EMG) to determine if the herniated disc is impinging on a nerve root. Diagnostic tests are often based on quantitative markers and the quality of the tests is typically measured by means of their sensitivity and specificity. Sensitivity is the true positive probability, i.e., the probability that an ill patient is identified as being ill. On the other hand, specificity is the true negative probability, which means the probability that a healthy patient is identified as such. Of course, high values of sensitivity and specificity are desired. However, when using a marker, moving its threshold to identify an ill patient as well as possible increases the test's specificity but decreases its sensitivity.

Denoting by $\Upsilon_0$ the marker for the diagnostic test for healthy people and by $\Upsilon_1$ the marker for diseased, and given a threshold value, $u$, for the diagnostic test, the sensitivity is $P(\Upsilon_1 > u)$ and the

Verónica Bolón-Canedo and Amparo Alonso-Betanzos (eds.), *Microarray Bioinformatics*, Methods in Molecular Biology, vol. 1986, https://doi.org/10.1007/978-1-4939-9442-7_11, 

specificity is $P(\Upsilon_0 \leq u)$. In this context, the receiver operating characteristic (ROC) curve is just the curve described by plotting the true versus false positive probabilities as a function of the threshold. ROC curves were first developed by electrical and radar engineers during World War II, but they were soon introduced in psychology and since then very much used in medicine (see the seminal work by Green and Swets [1], Swets and Pickett [2] and Hanley [3]). Mathematically, the ROC curve is defined as $\{(P(\Upsilon_0 > u), P(\Upsilon_1 > u)) : u \in \mathbb{R}\} = \{(1 - F_0(u), 1 - F_1(u)) : u \in \mathbb{R}\}$, with $F_0$ (respectively, $F_1$) the cumulative distribution function (cdf) of $\Upsilon_0$ (respectively, $\Upsilon_1$). Alternatively, by considering $t = 1 - F_0(u)$ the ROC curve can be parameterized by $\{(t, 1 - F_1(F_0^{-1}(1 - t))) : t \in [0,1]\}$.

In the extreme case where the distributions of the diagnostic marker for healthy and diseased people are the same (i.e., $F_0 = F_1$) the ROC curve results in the diagonal line for the first quadrant: $\{(t, t) : t \in [0, 1]\}$, with no more diagnostic power than a random choice. In general, the ROC curve is above the diagonal and the closer to the point $(0, 1)$, the better the diagnostic power. A common index that can be computed with the ROC curve is the area under the curve (AUC). This index is smaller than or equal to one (the area of the unit square). The larger the AUC, the better the diagnostic power. The extreme case of no diagnostic power at all (i.e., random choice for diagnosis) corresponds to the value 0.5 for AUC.

Statistical properties and extensions of ROC curves have been extensively studied, *see* [4–10] for related work. For readers interested in relevant publications concerning ROC curves for different medical problems we refer to [11–15], among many others.

## 2 ROC Analysis for Microarray Data

Over the past two decades ROC analysis has been applied to diagnostic tests based on microarrays. In this section we will concentrate on the papers [16–18].

Pepe, Longton, Anderson, and Schummer (*see* [16]) consider several statistical methods to rank genes (or proteins) in regard to differential expression between tissues, aiming to distinguish between cancerous and normal organ tissues. They propose to use AUC and a version of it, the pAUC, the partial area under the ROC curve between 0 and a certain value $t_0 \in (0, 1)$. The authors propose to select the value $t_0$ based on false positive rates that are acceptable in practice. For instance $t_0$ could be chosen as the maximal acceptable false positive rate. In general, very small values for $t_0$ are required in cancer screening; however, with small number of tissue samples, estimation of $pAUC(t_0)$ is not very accurate. For

the selected value, $t_0$, they suggest to use $ROC(t_0)$ and $pAUC(t_0)$ to rank genes. Unlike classical settings in which standard errors and $p$-values are considered, the authors suggest using the "selection probability function" for quantifying sample variability. For a gene $g$ and a number of top genes, $k$, the selection probability function is defined as $P(Rank(g) \leq k)$. The bootstrap method is used to estimate this function. The authors propose a method to calculate the sample size based on the selection probability function. The methods presented in [16] are used to analyze a real data set of gene expression arrays of 23 normal and 30 ovarian cancer tissues.

Tsai and Chen (*see* [17]) depart from [16] and use measures as $ROC(t_0)$, $pAUC(t_0)$, and $AUC$, with an arbitrary choice of the threshold, $t_0 = 0.1$, in practice. These authors propose to use $p$-values, estimated based on permutation tests, to calculate scores for ranking genes. With this approach, selecting the significant genes is equivalent to testing a large number of hypotheses, which brings the issue of multiple testing (*see* Subheading 3.2).

Berrar and Flatch (*see* [18]) discuss limitations of the ROC curve in the context of microarray data. The authors point out the drawbacks of ROC curves that are not convex and how to convexify non-convex ROC curves. The authors also argue that the cost curves may be an interesting alternative to ROC curves when the cost of the two possible misclassification errors can be quantified. The difference between classifying and ranking is also stressed in [18]. In that sense, the concept of ROC curve and AUC is better suited for measuring the ability of rankers than classifiers. Classifiers and rankers optimize different loss functions. Whereas classifiers aim at minimizing classification errors, rankers try to minimize the number of ranking errors. Another important caution when using ROC curves, mentioned in [18], concerns models derived from different training data sets. This is often done when performances are compared with published results. In those cases, the training sets are rarely the same when using different data, and sampling mechanisms turn out to be of vital importance. The fallacy of the undistributed middle is also presented to show how $AUC = 0.5$ does not necessarily imply that the classifier behaves like a random guessing. This is illustrated with some examples where classification may be even perfect but the behavior of the quantitative marker is not monotonic in terms of positive and negative instances, but two thresholds are needed instead to classify between normal (in the interval $(u_0, u_1)$) and diseased (in the set $(-\infty, u_0] \cap [u_1, \infty)$). In connection with this, the early retrieval problem is considered and it is shown how AUC may be useless when one is interested in just correctly classifying some top ranked cases (see also the concentrated AUC proposed by Swamidass et al. [19]). Multiple testing issues are also important when using AUC and its $p$-values in gene selection, when hundreds or thousands of genes are investigated. The authors also point out the importance

of using confidence intervals, rather than $p$-values, recommending the use of the bootstrap given the non-normality of the AUC distribution. All these points are discussed by means of illustrative examples coming from real-world studies.

## 3 ROC Analysis with Covariates

It is quite common in clinical practice that multiple diagnostic tests are performed on an individual. When multiple disease markers are available, a reasonable way to proceed to diagnose disease is to combine the information of those markers. Some of the very first papers that proposed to use linear combinations of markers in order to optimize diagnostic accuracy are [20] and [21]. Pepe and Thompson (*see* [20]) consider maximizing AUC and propose a distribution-free rank-based approach for optimizing the area under the ROC curve. The method is shown to be efficient in simulation studies and may be extended to deal with partial areas under the ROC curve, which allows one to focus on relevant regions of the ROC curve concerning clinical practice. These authors use this approach to study two cancer data sets, one involving serum antigen markers for pancreatic cancer and the other one concerning longitudinal prostate-specific antigen data. Following these ideas, Pepe, Cai, and Longton (*see* [21]) compare this linear combination approach maximizing AUC with classical maximum likelihood criteria dealing with multiple markers. Simulation studies performed by these authors suggest that AUC-based classification scores have a similar performance to logistic likelihood-based scores when the logistic regression model holds. On the other hand, the performance of the proposed method can be much better than logistic regression when this parametric model does not hold. This is illustrated by analyzing a data set from a proteomics biomarker study. The authors conclude that maximizing the AUC (rather than the likelihood) should be considered when the goal is to derive a marker combination score for classification or prediction.

The two previous papers do not consider applications to microarrays. In fact, this approach has a high computational burden when the number of markers is larger than two and no specific method is mentioned to deal with the problem of a large number of markers. This is extremely important for microarray data. Two of the papers recently published that are useful to overcome the computational limitations of the previous approach are [22] and [23]. Liu, Liu, and Halabi (*see* [22]) propose a min–max method which is a variant of the one in [21]. Simulation results by these authors show that the min–max method is more robust against distributional assumptions. The method is applied to a growth-related hormones data set from the Growth and Maturation in Children with Autism or

Autistic Spectrum Disorder Study (Autism/ASD Study). Although the method is computationally efficient for high-dimensional data, an important drawback is that when the dimension tends to infinity, the minimum sensitivity tends to 0 and the maximum sensitivity tends to 1, while the minimum specificity tends to 1 and the maximum specificity tends to 0. Another variation of [21] is the method proposed by Kang, Liu, and Tian (*see* [23]). These authors propose a nonparametric stepwise method and use a leave-one-pair-out cross-validation method to evaluate the performance of different linear combination methods in terms of AUC. Although this method combines the markers in a computationally efficient way, the fact that all the markers are combined makes the method not very well suited for microarray data. The authors apply this approach to a Duchenne muscular dystrophy data set.

### 3.1 LASSO-Like Techniques

Problems arising when the number of covariates is large are nowadays common in the statistical practice. In the context of linear regression with a large number of covariates, Tibshirani (*see* [24]) proposed the LASSO (least absolute shrinkage and selection operator) for multiple linear regression. Its key idea is to minimize an $L_1$-penalized version of the classical sum of squares. Denoting the data by $(\boldsymbol{X}_i, \Upsilon_i)$, $i = 1, \ldots, n$, where $\boldsymbol{X}_i$ is the $i$th observation of the column vector of explanatory covariates and $\Upsilon_i$ is the $i$th observation of the response variable, the LASSO simply minimizes, in the parameter vector $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\top$, the expression:

$$\sum_{i=1}^{n} (\Upsilon_i - \boldsymbol{\beta}^\top \boldsymbol{X}_i)^2 + \lambda \sum_{j=1}^{p} \left|\beta_j\right|,$$

where $\lambda > 0$ is a penalty term. An alternative way of formulating LASSO is to minimize $\sum_{i=1}^{n} (\Upsilon_i - \boldsymbol{\beta}^\top \boldsymbol{X}_i)^2$ subject to $\sum_{j=1}^{p} \left|\beta_j\right| \leq t$. When $t$ is large, the solution of this optimization problem is very close to the classical ordinary least squares estimator, while for small values of $t$, LASSO gives a solution with plenty of coefficients $\beta_j = 0$. An efficient iterative algorithm to find the LASSO solution has been proposed also in [24].

While LASSO is well suited for regression models with continuous response, the method must be adapted to the context of classification. A possible way to do this consists in using optimal scoring (*see* [25] and [26]). Ghosh and Chinnaiyan (*see* [27]) use optimal scoring ideas to carry out LASSO for minimizing a penalized version of AUC. Denoting by $\boldsymbol{X}_i$ the $i$th observation of the gene expression profile vector (i.e., $X_{ij}$ is the gene expression measurement of the $j$th gene, $j = 1, \ldots, p$, for the $i$th observation in the sample, $i = 1, \ldots, n$) and by $g_i \in \{0, 1\}$ the indicator of a diseased patient, the optimal scoring problem finds the vector of

coefficients $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_p)^\top$ and the scoring map $\theta : \{0,1\} \to \mathbb{R}$ that minimize the following average squared residual:

$$ASR = n^{-1} \sum_{i=1}^{n} (\theta(g_i) - \boldsymbol{\eta}^\top \boldsymbol{X}_i)^2.$$

An equivalence between LASSO and support vector machines (SVM) allows Ghosh and Chinnaiyan to carry out their method using standard software. These authors applied their method to a data set coming from a prostate cancer study.

### 3.2 Multiple Testing Issues

Regular $p$-values cannot be directly used to quantify the likelihood of hundreds or thousands of simultaneous hypotheses, which is often the case when dealing with high-dimensional data. This is because many features are tested at the same time and the probability that at least one null hypothesis is incorrectly rejected is not so small. This is known as the multiple comparisons or multiple testing problem. The most widely used approach to multiple testing is to define a procedure and then control an informative error rate for it. This means adapting the procedure to guarantee an error rate below a predefined value. The procedures are typically flexible through parameters or cutoffs that allow one to control specificity and sensitivity. An example of a procedure is: (a) compute a $p$-value for each gene; (b) consider significant all genes with $p$-values smaller than some value $\alpha$. Observe that changing the value of $\alpha$ permits to adjust specificity and sensitivity.

The family-wise error rate (FWER) is the probability of making at least one type I error in the family of hypotheses to be tested (*see* [28] and [29]). In the context of microarray data, FWER could be regarded as the probability that at least one non-influential gene is incorrectly detected as significant among all the tested genes. Controlling FWER is an important issue to ensure simultaneous correctness of a set of inferences as to guarantee a correct overall decision. However, FWER is often in practice a too restrictive way to control for multiple testing. If the number of genes tested is large, as it is the case in microarray experiments, the FWER approach tends to discard all but a few genes showing extreme differential expression. The false discovery rate (FDR) is an alternative method for accounting the rate of type I errors in null hypothesis testing when conducting multiple comparisons. FDR procedures control the expected proportion of "discoveries" (rejected null hypotheses) that are false (incorrect rejections). For microarray data, FDR controls the number of non-influential genes that are incorrectly detected as significant, when testing for all genes. FDR provides less stringent control of type I errors compared to FWER procedures, which control the probability of at least one type I error. Thus, FDR procedures have greater power, at the cost of an increased numbers of type I errors. Benjamini and

Hochberg (*see* [30]) and Benjamini and Yekutieli (*see* [31]) proposed very popular choices for FDR procedures. They essentially consist in computing the $p$-values for all the hypotheses to be tested, sorting and comparing them with a sequence of thresholds depending on the rank of each sorted $p$-value. It has been argued (*see* [32]) that, under general conditions, these procedures cannot control the FDR conditional to having rejected one or more hypotheses.

There exist several methods in the literature specifically designed for multiple testing when dealing with microarray data. Among them we mention the following papers: [33–37].

Tusher, Tibshirani, and Chu proposed the SAM approach (*see* [33]) as a method for estimating FDR, rather than controlling it, with a fixed rejected region.

Hsueh, Chen, and Kodell (*see* [34]) point out that the error rate of FWER or FDR procedures is usually lower than the specified level. These authors proposed five methods to estimate the number of true null hypotheses, and then used this estimated number to improve the power of FWER or FDR. Monte Carlo simulations showed that the so-called lowest slope and mean of differences methods appear to perform the best. These two methods control the FWER properly when the number of false null hypotheses is small. These methods are applied to a data set from a toxicogenomic microarray experiment.

Tsai, Hsueh, and Chen (*see* [35]) consider a convolution of two binomial distributions for the number of rejections and a noncentral hypergeometric distribution for the conditional distribution of the number of false rejections given the number of rejections. The authors used these distributions as well as some other more complex ones valid for equicorrelation and proposed five FDR probability error measures, which were evaluated by simulation.

In the context of cDNA, Delongchamp, Bowyer, Chen, and Kodell (*see* [36]) propose to use a plot of the observed $p$-values versus their expectation under a uniform $[0, 1]$ distribution to estimate the number of true null hypotheses (true influential genes). Using this estimate, the false positive rates and false negative rates can be estimated for any $p$-value cutoff. The authors proposed to select the cutoff by a decision-theoretic method similar to methods developed for ROC curves. They also applied these procedures to two functional genomics studies that were designed to assess a treatment effect.

Chen, Wang, Tsai, and Lin (*see* [37]) considered four commonly used statistics ($t$, SAM, Mann–Whitney $U$, and the $M$-statistic) to compute the $p$-values for gene ranking. FWER- and FDR-controlling procedures were used to select a limited number of genes, and a ROC approach to select a larger number of genes for assigning the significance level. A colon cancer data set was used to illustrate different gene ranking and significance level assignment

methods for applications to genomic/genetic profiling studies. The use of $p$-values or FDR probability, as the primary criterion, and then the fold-change as a surrogate measure of biological significance for gene selection is also discussed by these authors.

## 4 Conclusions

The use of ROC curves for microarray data is considered in this chapter. In general, ROC curves are very useful tools to measure the discrimination power of a binary classifier and, as a consequence, ROC analysis is a key element for evaluating how good a genetic marker can be for microarray data. The area under the ROC curve (AUC) and the partial area under the ROC curve (pAUC) are natural indices to measure the power of a marker to classify between healthy and ill patients (or normal and cancerous tissues). It is often the case that a vast number of genetic markers that are not of interest to the investigator may be present in the microarray. As a consequence, the selection, ranking, and combination of genetic markers are extremely important issues for microarray data. These issues can be addressed using a ROC analysis view, based on the AUC and pAUC. LASSO and SVM techniques are very useful to combine genes and to select the genes that are useful for discrimination. On the other hand, since the number of involved genes is high, multiple testing issues need to be handled using algorithms that control the false discovery probability and the expected number of false discoveries, like FDR and FWER. In summary ROC analysis for high-dimensional data is a research line with a promising future and plenty of room for applications to microarray data.

### References

1. Green DM, Swets JA (1966) Signal detection theory and psychophysics. Wiley, New York
2. Swets JA, Pickett RM (1982) Evaluation of diagnostic systems: methods from signal detection theory. Academic Press, New York
3. Hanley JA (1989) Receiver operating characteristic (ROC) methodology: the state of the art. Crit Rev Diagn Imaging 29(2):307–335
4. Pepe MS (1997) A regression modelling framework for receiver operating characteristic curves in medical diagnostic testing. Biometrika 84:595–608
5. Metz CE, Herman BA, Shen JH (1998) Maximum likelihood estimation of receiver operating characteristic (ROC) curves from continuously-distributed data. Stat Med 17:1033–1053
6. Alonzo TA, Pepe MS (2002) Distribution-free ROC analysis using binary regression techniques. Biostatistics 3:421–432
7. Pepe MS, Longton G (2005) Standardizing markers to evaluate and compare their performances. Epidemiology 16:598–603
8. Gu W, Pepe MS (2009) Estimating the capacity for improvement in risk prediction with a marker. Biostatistics 10:172–186
9. Huang Y, Pepe MS (2009) Biomarker evaluation and comparison using the controls as a reference population. Biostatistics 10:228–244
10. Pepe MS, Longton G, Janes H (2009) Estimation and comparison of receiver operating characteristic curves. Stata J 9:1–16
11. Mari JD, Williams P (1985) A comparison of the validity of 2 psychiatric screening

questionnaires (GHQ-12 and SRQ-20) in Brazil, using relative operating characteristic (ROC) analysis. Psychol Med 15(3):651–659

12. Greiner M, Sohr D, Gobel P (1995) A modified ROC analysis for the selection of cutoff values and the definition of intermediate results of serodiagnostic tests. J Immunol Methods 185(1):123–132
13. Rankinen T, Kim SY, Perusse L, Despres JP, Bouchard C (1999) The prediction of abdominal visceral fat level from body composition and anthropometry: ROC analysis. Int J Obes 23 (8):801–809
14. Chan HP, Sahiner B, Helvie MA, Petrick N, Roubidoux MA, Wilson TE, Adler DD, Paramagul C, Newman JS, Sanjay-Gopal S (1999) Improvement of radiologists' characterization of mammographic masses by using computer-aided diagnosis: an ROC study. Radiology 212(3):817–827
15. Baker SG (2003) The central role of receiver operating characteristic (ROC) curves in evaluating tests for the early detection of cancer. J Natl Cancer Inst 95:511–515
16. Pepe MS, Longton G, Anderson GL, Schummer M (2003) Selecting differentially expressed genes from microarray experiments. Biometrics 59:133–142
17. Tsai CA, Chen JJ (2004) Significance analysis of ROC indices for comparing diagnostic markers: applications to gene microarray data. J Biopharm Stat 14(4):985–1003
18. Berrar D, Flach P (2012) Caveats and pitfalls of ROC analysis in clinical microarray research (and how to avoid them). Brief Bioinform 13 (1):83–97
19. Swamidass SJ, Azencott CA, Daily K, Baldi P (2010) A CROC stronger than ROC: measuring, visualizing and optimizing early retrieval. Bioinformatics 26(10):1348–1356
20. Pepe MS, Thompson ML (2000) Combining diagnostic test results to increase accuracy. Biostatistics 1(2):123–140
21. Pepe MS, Cai T, Longton G (2006) Combining predictors for classification using the area under the receiver operating characteristic curve. Biometrics 62(1):221–229
22. Liu C, Liu A, Halabi S (2011) A min-max combination of biomarkers to improve diagnostic accuracy. Stat Med 30(16):2005–2014
23. Kang L, Liu A, Tian L (2016) Linear combination methods to improve diagnostic/prognostic accuracy on future observations. Stat Methods Med Res 25(4):1359–1380
24. Tibshirani R (1996) Regression shrinkage and selection via the LASSO. J R Stat Soc Ser B 58 (1):267–288
25. Hastie T, Tibshirani R, Buja A (1994) Flexible discriminant analysis by optimal scoring. J Am Stat Assoc 89(428):1255–1270
26. Hastie T, Buja A, Tibshirani R (1995) Penalized discriminant analysis. Ann Stat 23 (1):73–102
27. Ghosh D, Chinnaiyan AM (2005) Classification and selection of biomarkers in genomic data using LASSO. J Biomed Biotechnol 2:147–154
28. Hochberg Y, Tamhane AC (1987) Multiple comparison procedures. Wiley, New York
29. Westfall PH, Young SS (1993) Resampling-based multiple testing. Wiley, New York
30. Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. J R Stat Soc Ser B 57:289–300
31. Benjamini Y, Yekutieli D (2001) The control of the false discovery rate in multiple testing under dependency. Ann Stat 29(4):1165–1188
32. Zaykin DV, Young SS, Westfall PH (2000) Letter to editor. using the false discovery rate in the genetic dissection of complex traits. Genetics 154:1917–1918
33. Tusher VG, Tibshirani R, Chu G (2001) Significance analysis of microarrays applied to the ionizing radiation response. Proc Natl Acad Sci USA 98:5116–5121
34. Hsueh HM, Chen JJ, Kodell RL (2003) Comparison of methods for estimating the number of true null hypotheses in multiplicity testing. J Biopharm Stat 13(4):675–689
35. Tsai CA, Hsueh HM, Chen JJ (2003) Estimation of false discovery rates in multiple testing: application to gene microarray data. Biometrics 59(4):1071–1081
36. Delongchamp RR, Bowyer JF, Chen JJ, Kodell RL (2004) Multiple-testing strategy for analyzing cDNA array data on gene expression. Biometrics 60(3):774–782
37. Chen JJ, Wang SJ, Tsai CA, Lin CJ (2007) Selection of differentially expressed genes in microarray data analysis. Pharmacogenomics J 7:212–220

# Chapter 12

# Missing-Values Imputation Algorithms for Microarray Gene Expression Data

**Kohbalan Moorthy, Aws Naser Jaber, Mohd Arfian Ismail, Ferda Ernawan, Mohd Saberi Mohamad, and Safaai Deris**

## Abstract

In gene expression studies, missing values are a common problem with important consequences for the interpretation of the final data (Satija et al., Nat Biotechnol 33(5):495, 2015). Numerous bioinformatics examination tools are used for cancer prediction, including the data set matrix (Bailey et al., Cell 173 (2):371–385, 2018); thus, it is necessary to resolve the problem of missing-values imputation. This chapter presents a review of the research on missing-values imputation approaches for gene expression data. By using local and global correlation of the data, we were able to focus mostly on the differences between the algorithms. We classified the algorithms as global, hybrid, local, or knowledge-based techniques. Additionally, this chapter presents suitable assessments of the different approaches. The purpose of this review is to focus on developments in the current techniques for scientists rather than applying different or newly developed algorithms with identical functional goals. The aim was to adapt the algorithms to the characteristics of the data.

**Key words** Missing-values imputation, Gene expression data, Microarray, Cancer Informatics, Computational intelligence

## 1 Introduction

In the microarray method, scientists use several important tools to observe the appearance of different genes in a particular creature [1]. In genetic analyses, microarray technology is especially important for microdimension chips, which contain a huge number of genes that can be used for extensive investigations of gene expression. This technology allows for the development of sample information, to which statistics can be applied for the detection of gene expression and gene regulation. In studies on cancer organization, the detection of appropriate genes for the analysis can be achieved with the microarray technique. In addition, the technique is also used to examine the effects of drugs in cancer diagnosis [2, 3]. This method has also proven to be useful in other fields, such as

Verónica Bolón-Canedo and Amparo Alonso-Betanzos (eds.), *Microarray Bioinformatics*, Methods in Molecular Biology, vol. 1986, https://doi.org/10.1007/978-1-4939-9442-7_12, 

immunology, microbiology, and virology [4]. By identifying testing samples from altered diseases, the microarray method can be improved to investigate different classes [5].

In the field of bioinformatics microarray, data investigations are challenging because there are uncharacterized variables that require interpretation [6]. In particular, problems may be encountered in gene expression analysis because of missing values [7]. Missing values are considered to be inconsequential when they occur at a rate of less than 1% and controllable at 1–5%. However, a rate of 5–15% requires sophisticated methods to handle the imputation, whereas a rate exceeding 15% greatly affects the estimate or interpretation. Numerous explanations for missing values have been reported, including the presence of artifacts on the microarray, low resolution, and hybridization failures [8].

Other investigations have shown that data with missing values can greatly affect an analysis and hamper the detection of differently expressed genes, supervised and unsupervised classification of genes, and the creation of gene regulation networks [9]. Furthermore, missing values in data have shown negative consequence on algorithms, such as singular value decomposition (SVD), support vector machines, and principal component analysis (PCA) [10].

The imputation of missing values to replace the probability of informative genes through the assortment is important. The issue can be resolved by recent developments in algorithms for missing values approximation; thus, the detection of useful genes will be protected. This is vital for the primary detection of genes that are specific targets of a particular class [11]. Therefore, missing-values approximation is considered to be an effective and low-cost approach. By using computational techniques, the missing values in data can be estimated without the need to repeat all microarray experiments.

Previous studies have established that inclusive detection or data scrutiny can be used for missing-values imputation [12]. Several refined mathematical models have been established and were proven useful in different biomedical research fields [13]. In all of these methods, the absent values are estimated from the non-missing data set. In certain circumstances for gene expression data, the missing values are estimated from incorrect values, which affects the detection of disease-related genes. Hence, the probability of dropping useful genes is high and the general ranking of significant genes is disturbed [14]. Microarray experiments have resulted in many newly identified gene expressions Thus, they may not contain typical errors from repeating the microarray investigation.

For the development of cancer prediction algorithms, much data exist on cancer patients. However, pre-processed raw data may have missing values from the use of outdated approaches, which are frequently centered on *k*-nearest neighbor. The replication of gene

expression profiles allows for development in the pre-processing stage; this has been extensively used to estimate cancer, such as colon cancer and brain cancer.

Early methods to address missing values included the elimination of the entire row that contained missing values, imputing missing values with median or regular row values, and changing the missing values to be zero [15]. However, replacing missing values with average values or zero is not the best approach because it may introduce biases, as the association of the data is not counted.

## 2 Review of Missing-Values Imputation Algorithms

In early imputation approaches, software was used for the imputation of microarray data. The data centered on statistical scrutiny and natural evidence of the data was passed over. In current imputation approaches, imputation procedures were designed to use the information from the microarray data as an advantage [16].

The approaches to missing-values imputation can be divided into three categories: parameter estimation, pairwise deletion, and imputation techniques [17]. There are four altered approaches to deal with missing values [18]: k Nearest Neighbor (KNN) imputation (KNNI), median imputation (MDI), mean imputation (MI), and case deletion (CD). In CD, all cases or instances with missing values are removed. In MI, the missing values of an individual are exchanged by computing the mean based on all known values of attributes. The median is used for the MDI to declare consistency instead of meaning because the mean is influenced by the presence of outliers. The KNNI approach uses a distance function in which the similarity of two instances is determined and established on the illustrations that are furthermost parallel to the instance of attention, and the missing values are imputed [19].

The imputation techniques for missing values can be characterized into two principal types: generic statistical approaches and application-specific alterations. Generic statistical techniques include mean, hot deck, model-based, multiple, and cold deck imputations. The quality matters and investigational projects are taken into account when imputing gene expression data.

The investigations of current algorithms can be classified into four key types by the type of information used: global, local, hybrid, and knowledge-assisted approach. The algorithms are categorized and listed in Table 1.

### 2.1 Global Approach

In a global approach, the algorithms execute missing-value approximations based on global correlation material resulting from the whole data matrix. The imputation becomes less accurate when the algorithms take up the existence of a global covariance organization

**Table 1**
**List of missing-value imputations by category**

| Algorithm | Year | Category |
|---|---|---|
| SVDimpute | 2001 | Global |
| BPCA | 2003 | Global |
| MICE-CART | 2010 | Local |
| KNNimpute | 2001 | Local |
| GMCimpute | 2004 | Local |
| LLSimpute | 2005 | Local |
| SLLSimpute | 2008 | Local |
| CMVE | 2005 | Local |
| AMVI | 2008 | Local |
| ABBA | 2010 | Local |
| LinCmb | 2005 | Hybrid |
| HPM-MI | 2015 | Hybrid |
| FCM+GA | 2015 | Hybrid |
| HPF | 2015 | Hybrid |
| AR-ANN | 2015 | Hybrid |
| POCSimpute | 2006 | Knowledge |
| GOimpute | 2006 | Knowledge |
| HAIimpute | 2008 | Knowledge |

in all gene samples, as well as if the genes have identical structures. Algorithms based on a global approach include SVD [20] and Bayesian principal component analysis [21].

### 2.2 Local Approach

In a local approach, the algorithms show only a local correspondence arrangement for computing missing-value imputations within the data sets. The subgroups of genes that show high correlation through the genes are used to calculate the missing values in the gene. Some well-known algorithms that use this procedure include the local least square imputation (LLSimpute) and $K$ nearest neighbor (KNN). MICE-CART is a nonparametric method that uses multiple imputations by chained equations (MICE) and classification and regression trees (CART) [22].

Multiple imputations have also been performed through chained equations using sequential regression trees as the conditional models [23]. Complex relationships of the data are captured and applied to decrease the practice of limitation and imputing

tuning. For target genes with missing values, the *k* nearest reference genes are used in KNN to impute the missing values with pairwise information. KNN imputation works very well when robust correlations are found between genes in the data.

Gaussian mixture clustering (GMC) is able to use extra global correlation information even though it is considered to be a local approach algorithm [24]. In this algorithm, the data is clustered into *S* components. Gaussian mixtures using the *S* estimates of the missing values and EM algorithms are able to obtain the final estimated missing values. GMC uses the local correlation information through a mixture of components.

A local least squares imputation uses multiple regression models to impute missing values [25]. This method has been recognized to be marginally competitive with KNN imputation and considerably more difficult than Bayesian principal component analysis (BPCA). Sequential LL Simpute (SLLSimpute) is an extension of this method [26].

The Simpute algorithm implements the imputation sequentially by establishing the minimum missing rates from the gene. The imputed genes are then recycled for the imputation of other genes. Because of the reusability of the genes with missing values, SLLS imputation shows better performance than LL Simpute.

To improve the final estimation, the collateral missing value imputation (CMVE) technique uses the idea of numerous similar estimations of missing values [27]. For many datasets involving ovarian cancer and yeast sporulation time series data, CMVE [27] has been able to produce better accuracy in normalized RMS error (NRMSE) than BPCA, KNN, and L Simpute. By using Monte Carlo simulation for the determination of an ideal number of reference genes *K,* ameliorative missing value imputation (AMVI) has enhanced CMVE [28]. A strong dependency among observations is displayed by time series expression profiles.

For binary matrices, an adaptive bicluster-based approach (ABBA) is used as a missing-values estimator [29]. The complications of the algorithm itself have been decreased, whereas the amount of parameter alterations that can be achieved is increased. When the rate of missing values is much higher than normal, the algorithm has been verified to be superior to KNN [20].

### 2.3 Hybrid Approach

For heterogeneous data sets, local correlation among genes is dominant. In such cases, local imputation methods, such as KNN or LL Simpute, perform better compared with BPCA or SVD impute. This is because the correlation structure of the data affects the performance of the imputation systems. However, for homogeneous data, global methods such as BPCA and SVD would work better by capturing global correlation material in the data.

There are numerous hybrid methods for missing-values imputation, such as hybrid prediction model with missing value

imputation (HPM-MI). By using the best imputation methods, data quality can be considerably enhanced. HPM-MI uses 11 different missing-values imputation methods, combining *K*-mean clustering with Multilayer Perceptron and then selecting the best clusters among the results. Class labels of given data are validated using *K*-means clustering. This is a hybrid prediction model for medical data. After extensive examination of 11 imputation methods, the best imputation technique is used [27].

For gene expression, specific linear topology can be used. This topology has a state of transition and self-transitions to the next state. At the start of each chain, there is a special start state. The results can be interpreted as an expression close to background level, which is shown by an expression value near zero; a value above zero shows over-expression or up-regulation, whereas a value below zero shows under-expression or down-regulation. For example, consider a linear Hidden Markov Model (HMM) with two emitting states: the first one with a zero mean emission and the second with a one mean emission; this typically shows up-regulation prototypical behavior [30].

FCM+GA is an acronym for Fuzzy C-means based imputation + genetic algorithm**.** This approach is based on inductance loop detector outputs. Using the resemblance among data vector-based data, the structure transformed into a matrix-based data pattern. Hence, a genetic algorithm is used to optimize the membership functions and plays a central role in the FC model. The Fuzzy C is a hybrid method. The Fuzzy C imputation method is combined with the genetic algorithm to estimate the missing values in data [31]. This method is mostly used in pattern recognition. It has the advantage of giving the best modeling results. However, the weighting exponent (m) is fixed to a conventional value of 2, which is not suitable for all applications.

The iterative search approach is used to get an optimal number of clusters and find the optimal single-output Sugeno-Type. A Fuzzy inference system (FIS) model is used to obtain the minimum least square error by improving the limits of the subtractive clustering algorithm among the actual data and the Sugeno Fuzzy model. The two methods are proposed when the number of clusters is the optimized weighting exponent (m). These two approaches of iterative search and the genetic algorithms were tested on the data from the original function. The Fuzzy models were found to have the least error among the real data and Fuzzy model [31].

Fuzzy c-means (FCM) allows the feature vector to belong to all clusters with a Fuzzy truth value (between 0 and 1). The algorithm assigns a feature vector to clusters based on the maximum feature vector weight over other clusters. HPF is a hybrid approach. To avoid the intervals determined by altered cluster material, this hybrid method can be used. It is helpful in improving the clustering performance [32]. This algorithm uses global optimization. To

calculate membership information, the cluster prototypes and gradient-based FCM are used. The HPF hybrid approach uses the global optimization capability of particle swarm.

Artificial Neural Networks (ANN) is another method used for missing-values imputation. It is able to handle nonlinearity issues. The input layer structure of ANN can be determined using an autoregressive (AR) model. A hybrid AR-ANN method can be used to analyze the imputation of missing values [33]. Before AR modeling, deletion was used to deal with missing values in wind speed time series data sets. This also shows the nonlinearity of wind speed data.

The use of a multilayer feed-forward back propagation neural network for time series forecasting is supported by the ANN toolbox in MATLAB software. To create the most appropriate ANN structure, the types of hidden and output layers and other requirements are necessary. Hence, the training functions and the transfer functions need to be determined. The transfer functions are tan-sigmoid that generate nonlinear outputs between −1 and +1. The log-sigmoid generates nonlinear outputs between 0 and 1, whereas the linear generates linear outputs between −1 and +1. It is important to select a suitable function to obtain the best results. The best training functions for back propagation algorithms are Levenberg-Marquardt and Bayesian regularization. To create an applicable ANN, the number of neurons must be measured correctly in the hidden layers [34].

### 2.4 Knowledge-Assisted Approach

A knowledge-assisted approach incorporates domain information or outside data into the missing-values imputation. The imputation accuracy is expressively improved with the use of domain knowledge associated with a data-driven method, particularly for data sets with a high missing rate, a small number of samples, and noise.

The correlation information between genes and arrays is exploited when the missing-values imputation for projection onto convex sets (POCS) is a flexible set with a theoretic structure [35]. POC Simpute performs local least square regression to capture gene-wise correlations. It also performs PCA imputation to capture array-wise correlation and synchronization loss, which restricts the squared power of the expression profiles.

Using POC Simpute, the best solution can be obtained irrespective of global or local correlation structures that are prevalent in the data. This is due to the final solution being constantly dominated by the smallest yet furthermost consistent constraint set though adequate larger yet less reliable constraint sets. Functionally correlated genes are likely to be expressed in a modular fashion through a higher degree of concerted responses to certain stimuli [36].

For gene function classification, gene ontology (GO) is a well-accepted standard [37]. GO has three independent ontologies that

describe a gene product in terms of related biological processes, molecular functions (MFs), and cellular components [38]. GO increases the imputation accuracy. This has been proven when the number of experimental situations is small, the ratio of the annotated genes is large, or the rates of missing values are high.

To improve the precision of missing-values approximation, histone acetylation information-aided imputation (HAIimpute) combines histone acetylation information into KNNimpute and LLSimpute [27]. HAIimpute uses the mean expression of genes from all clusters to form the pattern expression. By fitting a linear regression model, the missing values are then found among the gene and pattern expressions. The final estimations of the missing values are assumed by a convex combination of linear regression imputations and secondary imputations using KNNimpute and LLSimpute. The experimental results proved that the imputed genes of HAIimpute show better correlation with the original complete genes compared with KNNimpute or LLSimpute.

## 3 Performance Evaluation

An assessment of an algorithm's imputation results is a critical step to determine the algorithm's reliability, performance, and accuracy. For missing-values imputation, there are two main validation types: internal validation and external validation. In internal validation, the performance indices are computed among the imputed and the known original values to validate the algorithms. This validation also uses information from the dataset. For external validation, the validation is prepared by following biological analysis to assess the imputation effects. External validation usually uses the knowledge assembled externally rather than internal data information. Table 2 presents a list of comparative validation methods.

### 3.1 Internal Validation

By computing the normalized root mean square error (NRMSE), the missing-values imputation algorithms can be validated. Lower NRMSE values indicate that the imputation algorithm is more precise.

The NRMSE is defined as follows:

$$\mathrm{NRMSE} = \sqrt{\frac{\sum_{i=1}^{m}\sum_{k=1}^{n}\left(g_{ik}-\tilde{g}_{ik}\right)^2}{\sum_{i=1}^{m}\sum_{k=1}^{n}\left(g_{ik}\right)^2}} \tag{1}$$

Here, the $k$th experiment is for gene $g_i$, with $g_{ik}$ and $\tilde{g}$ denoting the true value and imputed value, respectively.

**Table 2**
**Performance evaluation methods for missing-values imputation algorithms**

| Validation type |
|---|
| Internal validation |
| NRMSE or its variants |
| Pearson Correlation |
| Preservation of differentially expressed genes |
| Preservation of prediction/classification problem |
| External validation |
| GO enrichment |
| Presence of biologically relevant genes |

### 3.2 External Validation

The validity of the imputation algorithms can be determined by external information, such as pathway information and functional annotations. For example, GO can be considerably enriched between genes. It is thus useful to characterize the functional roles of the genes in biological research. For each GO term $t$, the enrichment of the $P$-value can be calculated using the following formula for each cluster:

$$p = \sum_{i=k}^{\min(b,T)} \frac{\binom{T}{i}\binom{B-T}{b-i}}{\binom{B}{b}} \tag{2}$$

The number of genes in the cluster is represented by $b$ for the GO term $t$, with the cluster of genes represented by $K$. In the data set, the number of genes is $B$, and the number of genes with GO term $t$ is represented by $T$.

## 4 Limitations

There are many advantages and disadvantages of the algorithms, according to the datasets being used for each missing-values technique. The performance of missing-values imputation algorithms is considerably affected by a variety of factors, such as the missing-data mechanism, the correlation structure in the data, the percentage of missing values in the data, and distribution of missing entries in the data. Selecting the right algorithm may significantly boost the accuracy of the imputation results. However, there is no one imputation algorithm that shows best results in every situation. For low-entropy data sets, global methods such as SVDimpute and BPCA perform better. For high-entropy data sets, local methods such as LLSimpute and KNNimpute perform better.

Most studies have shown that missing values in microarrays are randomly missing through missing-values imputations. In practice, the missing values also tend to arise in a systematic manner. In a data matrix, the distribution of missing entries, such as missing data patterns, influence the imputation performance; thus, this needs to be considered in data analysis and algorithm design. A data matrix that contains a non-random distribution of missing values may result from different experimental conditions across the columns. In a microarray data matrix, each column comes from one experiment.

## 5 Conclusion

Many studies have reported that missing-values imputation is a common problem because microarray gene expression data may include missing values, which affects a study's results. Due to various experimental causes, gene expression profiling techniques, such as cDNA microarray technology, suffer from the problem of missing values.

In microarray data analysis, missing-values imputation is an essential pre-processing step. Many analyses require a complete data set. If the existing algorithms could be assessed, matched, and recognized instead of establishing new algorithms that are based on existing techniques, it would be a great achievement.

For optimal execution of an algorithm, suitable parameters and operating platforms could be identified and assessed using different algorithms. This could be used to avoid the development of new missing-values imputation algorithms. These algorithms are expected to perform similarly because the full functional capability of existing algorithms has not been fully explored.

## 6 Future Works

Many algorithms have been explored for missing-values imputation. There is a need to identify algorithms that provide accurate results and adapt to the features of data sets. An adaptive technique that can capture correlation information from both the local and global methods would be useful in many situations. As more experimental data from different domains becomes available, new imputation algorithms that can handle categorical data and mixed domain data sets with missing continuous information will be required.

## Acknowledgements

We would like to thank Universiti Malaysia Pahang for supporting this work under the RDU Grant, Grant number: RDU1703200 and RDU180344.

### References

1. Fehrmann RS, Karjalainen JM, Krajewska M, Westra H-J, Maloney D, Simeonov A, Pers TH, Hirschhorn JN, Jansen RC, Schultes EA (2015) Gene expression analysis identifies global gene dosage sensitivity in cancer. Nat Genet 47(2):115
2. Lima-Tenório MK, Pineda EAG, Ahmad NM, Fessi H, Elaissari A (2015) Magnetic nanoparticles: in vivo cancer diagnosis and therapy. Int J Pharm 493(1-2):313–327
3. Bailey MH, Tokheim C, Porta-Pardo E, Sengupta S, Bertrand D, Weerasinghe A, Colaprico A, Wendl MC, Kim J, Reardon B (2018) Comprehensive characterization of cancer driver genes and mutations. Cell 173 (2):371–385; e318
4. Criscuolo E, Spadini S, Lamanna J, Ferro M, Burioni R (2017) Bacteriophages and their immunological applications against infectious threats. J Immunol Res 2017:3780697
5. Salem H, Attiya G, El-Fishawy N (2017) Classification of human cancer diseases by gene expression profiles. Appl Soft Comput 50:124–134
6. Saeys Y, Inza I, Larrañaga P (2007) A review of feature selection techniques in bioinformatics. Bioinformatics 23(19):2507–2517
7. Satija R, Farrell JA, Gennert D, Schier AF, Regev A (2015) Spatial reconstruction of single-cell gene expression data. Nat Biotechnol 33(5):495
8. Lai H-H, Chuang T-H, Wong L-K, Lee M-J, Hsieh C-L, Wang H-L, Chen S-U (2017) Identification of mosaic and segmental aneuploidies by next-generation sequencing in preimplantation genetic screening can improve clinical outcomes compared to array-comparative genomic hybridization. Mol Cytogenet 10(1):14
9. Danaee P, Ghaeini R, Hendrix DA (2017) A deep learning approach for cancer detection and relevant gene identification. In: Pacific symposium on biocomputing 2017. World Scientific, pp 219–229
10. Larose DT, Larose CD (2014) Discovering knowledge in data: an introduction to data mining. Wiley, Hoboken, NJ
11. Quinn JJ, Chang HY (2016) Unique features of long non-coding RNA biogenesis and function. Nat Rev Genet 17(1):47
12. Gogoshin G, Boerwinkle E, Rodin AS (2017) New algorithm and software (BNOmics) for inferring and visualizing Bayesian networks from heterogeneous big biological and genetic data. J Comput Biol 24(4):340–356
13. Zomorrodi AR, Segrè D (2016) Synthetic ecology of microbes: mathematical models and applications. J Mol Biol 428(5):837–861
14. Hu W, Lin X, Chen K (2015) Integrated analysis of differential gene expression profiles in hippocampi to identify candidate genes involved in Alzheimer's disease. Mol Med Rep 12(5):6679–6687
15. Cressie N (2015) Statistics for spatial data. Wiley, Hoboken, NJ
16. Hira ZM, Gillies DF (2015) A review of feature selection and feature extraction methods applied on microarray data. Adv Bioinformatics 2015:198363
17. Lang KM, Little TD (2018) Principled missing data treatments. Prev Sci 19(3):284–294
18. Josse J, Husson F (2016) missMDA: a package for handling missing values in multivariate data analysis. J Stat Softw 70(1):1–31
19. Tsai C-F, Li M-L, Lin W-C (2018) A class center based approach for missing value imputation. Knowl-Based Syst 151:124–135
20. Garvey C, Meng C, Nagy JG (2018) Singular value decomposition approximation via Kronecker summations for imaging applications. arXiv preprint arXiv:180311525
21. Chatfield C (2018) Introduction to multivariate analysis. Routledge, New York
22. Tran CT, Zhang M, Andreae P (2016) A genetic programming-based imputation method for classification with missing data. In: European conference on genetic programming. Springer, pp 149–163
23. Shah AD, Bartlett JW, Carpenter J, Nicholas O, Hemingway H (2014) Comparison of random forest and parametric imputation models for imputing missing data using MICE: a CALIBER study. Am J Epidemiol 179(6):764–774

24. Bhattacharya S, Rajan V, Anand A (2017) Clustering high dimensional data using gaussian mixture copula model with lasso based regularization. Google Patents
25. Fox J (2015) Applied regression analysis and generalized linear models. Sage Publications, Thousand Oaks, CA
26. van der Loo M (2017) Simputation: simple imputation. R package version 02 2
27. Armina R, Zain AM, Ali NA, Sallehuddin R (2017) A review on missing value estimation using imputation algorithm. J Phys Conf Ser 892:012004
28. Rubinstein RY, Kroese DP (2016) Simulation and the Monte Carlo method, vol 10. Wiley, New York
29. Colantonio A, Di Pietro R, Ocello A, Verde NV (2010) ABBA: adaptive bicluster-based approach to impute missing values in binary matrices. In: Proceedings of the 2010 ACM symposium on applied computing. ACM, pp 1026–1033
30. Smart Richman L, Blodorn A, Major B (2016) An identity-based motivational model of the effects of perceived discrimination on health-related behaviors. Group Process Intergroup Relat 19(4):415–425
31. Naik B, Mahapatra S, Nayak J, Behera H (2017) Fuzzy clustering with improved swarm optimization and genetic algorithm: hybrid approach. In: Computational intelligence in data mining. Springer, pp 237–247
32. Qi S, Schmid F (2017) Hybrid particle-continuum simulations coupling Brownian dynamics and local dynamic density functional theory. Soft Matter 13(43):7938–7947
33. Shukur OB, Lee MH (2015) Imputation of missing values in daily wind speed data using hybrid AR-ANN method. Mod Appl Sci 9 (11):1
34. Kayri M (2016) Predictive abilities of bayesian regularization and Levenberg–Marquardt algorithms in artificial neural networks: a comparative empirical study on social data. Math Comput Appl 21(2):20
35. Gan S, Wang S, Chen Y, Chen X, Huang W, Chen H (2016) Compressive sensing for seismic data reconstruction via fast projection onto convex sets based on seislet transform. J Appl Geophys 130:194–208
36. van der Loo M, de Jonge E (2018) Statistical data cleaning with applications in R. Wiley, New York
37. Mi H, Huang X, Muruganujan A, Tang H, Mills C, Kang D, Thomas PD (2016) PANTHER version 11: expanded annotation data from gene ontology and reactome pathways, and data analysis tool enhancements. Nucleic Acids Res 45(D1):D183–D189
38. Aziz MF, Caetano-Anollés K, Caetano-Anollés G (2016) The early history and emergence of molecular functions and modular scale-free network behavior. Sci Rep 6:25058

# Chapter 13

# Computer Tools to Analyze Microarray Data

Giuseppe Agapito

## Abstract

Microarrays are broadly used in genomic analyses and find several applications in biology and medicine, providing a significant amount of data from a single experiment. Different kinds of microarrays are available which are identifiable by characteristics such as the type of probes, the surface used as support, and the method used for target detection. Although microarrays have been applied in many biological areas, their management, and investigation require advanced computational tools to speed up data analysis and at the same time make the interpretation of the results easier. To better deal with microarray datasets of large size, the development of analysis tools that are simple to use as well as able to produce accurate predictions, and of comprehensible models is essential. The tools have to provide an exhaustive collection of information to discriminate and identify SNPs, which are associated with the activity of particular genes affecting biological functions (e.g., a particular drug response), or involved in the development of complex diseases. The object of this chapter is to provide a review of software tools that are easy to use even for nonexperts of the domain, and that are able to efficiently deal with microarray data.

**Key words** Microarrays, Statistical analysis, Data mining, Genomics, Genotyping

## 1 Introduction

The discovery of the 3D structure of DNA by Watson and Crick [1] contributed to the rapid improvement of genotyping techniques. After the 3D structure of DNA was discovered initial efforts focused on sequencing a small fragment of pure RNA. In 1977 thanks to Sanger there was a major breakthrough in the sequencing methodology with the development of the Sanger's "Chain Termination" method, also known as dideoxy method [2]. The accuracy, robustness, and ease of use of Sanger sequencing has made it the most used technology to sequence DNA. Meanwhile, the formation of hybrid DNA molecules through the employment of heavy isotope-labeled DNA molecule has been used to shed light on DNA strand separation and recombination [3]. In [4] the authors illustrated and developed an application of hybridization technique can be applied to the DNA of a significant number of colonies of *Escherichia coli* characterized by different hybrid plasmids. The

Verónica Bolón-Canedo and Amparo Alonso-Betanzos (eds.), *Microarray Bioinformatics*, Methods in Molecular Biology, vol. 1986, https://doi.org/10.1007/978-1-4939-9442-7_13, 

proposed technique can quickly screen all the colonies to discover which hybrid plasmids include a specified DNA sequence or gene. Thus, the technique developed in [4] can be considered an initial example of a labeled probe used to identify complementary base pairing. Consequently, this method can be considered the first example of a microarray assay.

Nowadays, microarray technology has become one of the standard ways to study gene expression, single nucleotide polymorphisms (SNPs), and diagnosis of disease. The main strength of microarrays is the capability to compare the expression and regulation of thousands of genes in parallel for a single experiment helping physicians to figure out the harmful genes responsible for the disease. Before the automatization and miniaturization of microarrays, researchers have been limited to observations of smaller numbers of genetic units per single experiment and were able to evaluate only interacting genes on the smaller scale. Microarray technology is especially useful in the evaluation of gene expression patterns in complex disorders thanks to its ability to monitor the expression of the same genes in different samples at the same time, providing clues about interactions between multiple genetic units, unlike analyzing single genes per time. A microarray is composed of a solid surface (i.e., glass, silicon, nylon) with several DNA spots attached to it. Each spot contains a small sequence of DNA (gene) of interest called probe. A collection of probes which have the same nucleotide sequences is called a probeset which will assist in detecting the expression of a particular gene. Microarrays arrange biological samples in a two-dimensional space, where samples are placed within the spots and organized in columns and rows. This scheme allows for an efficient analysis of the genetic material in microarrays, making microarrays well suited for large-scale case–-control studies.

A general microarray study requires a considerable quantity of complementary DNA (cDNA) or oligonucleotide DNA sequences (spots) that are attached to the microarray's plate. The microarray is then reacted with two series of messenger RNA (mRNA) probes (cases and controls) that are treated with two different isotopes allowing to obtain two different colors of fluorescent for the two classes of probes. At the end of the hybridization process, the microarray is washed. Washing the microarray allows the probes that are not hybridized to be removed from the base. Thus, only the hybridized probes that are attached to the spots on the slide remain. After the washing, it is mandatory to read the microarray. Microarray reading happens through a scanner using a laser beam to generate an image of the intensity of all the spots. The intensity of the fluorescent signal from each spot is taken as a measure of the levels of the mRNA associated with the specific hybridized sequence at that spot. The image of all the spots is analyzed using special proprietary software that are able to convert the intensity

value of each spot into numerical information. This then makes possible to estimate the gene expression level in each spot as well as detect single nucleotide polymorphisms (SNPs) via genotype translation. Typically, a microarray experiment comprises the following phases:

- Microarray choice: This step comprises (a) the selection of the DNA sequences to be used in the analysis; (b) the methodology to attach the sequences onto the plate.
- Probe preparation and hybridization: First, it is necessary to extract some samples of RNA (i.e., healthy and cancerous biological tissues) from the samples under investigation, and purifying mRNA from each sample. Second, by using copy, clone, and reverse transcriptase (via polymerase chain reaction) the cDNA copies of mRNAs are synthesized and used in vitro transcription, then they are converted to cRNA and labeled with fluorescent solutions. Finally, the is mixture is placed on the microarray's probes. Complementary RNAs to the molecules on the microarray hybridize with the strands in the microarray.
- Microarray washing: At the end of the hybridization process, washing the microarray helps to highlight remarkably the hybridized probes (tightly chemically bound) remaining thus attached to the plate.
- Hybridization intensity reading: The last step is the intensity reading that is the result of the hybridization process. Reading is achieved by using a laser beam that shines on the slide causing a different level of fluorescence related to each cDNA component. Thus, the greater the length of hybridized segments, the higher the intensities.

At the end of the main steps listed above, before to extract actionable knowledge, it is mandatory to preprocess the intensity files. Intensity files contain noise due to the spots, that are often composed of imperfections due to irregular contours, artifacts, and low expression. The easiest initial cleaning attempt is to perform background correction. Background correction approach consists in subtracting the background estimate value directly from the spot intensity. In fact, it is often considered that the signal obtained as a result is the combination of the hybridization and the background signal (noise). Background noise may increase due to dust, fingerprints, hybridization problems, or residual effects from inadequate washing. Thus, many image analysis methods have been adapted to deal with the microarrays results read, to obtain reliable data with which conduct further analysis.

Users can export the preprocessed and cleaned data in order to perform further automatic or manual data analysis. Because microarray datasets are commonly very large, this makes manual analysis unfeasible, time consuming and error prone, thus the necessity of

highly automated data analysis tools arise. In order to extract biological relevant information contained in microarray datasets, a basic knowledge of the employed computational tools is therefore required for optimal experimental design and meaningful data analysis.

## 2 Type of Microarrays

Features such as the nature of the probe, the plate used, and the method to detect the target can be used to categorise microarrays. Thus, microarrays can be classified as, printed, in situ-synthesized, high-density bead, electronic, and suspension bead microarrays. The DNA sequence bound to the microarray support (i.e., glass, ceramic, and silico) is known as probe, while the sequence of DNA under investigation is called target. A microarray is organized as a big table composed of thousands of identical probes (a well-known sequence to whom the target sequence will bind through hybridization), whereas the target is an unknown fluorescently labeled sequence (complementary sequence). The sample sequences can be labeled by using radioactive isotope such as 33P, or fluorescent dyes such as phycoerythrin, *Cy3*, or *Cy5*. The hybridization between the labeled target and the probe results in an increase in fluorescence intensity that can be read by using specific optical scanners, which are able to evaluate the quantity of mRNA produced during the hybridization process. Due to the base pairing of the four nucleotides adenine, cytosine, guanine and, thymine, in short A, C, G, and T, for DNA and A, C, G, U (uracil) for RNA, the hybridization is a well-known and deterministic process. In particular, only complementary nucleotides can bind together, that is, A can bind with its complementary T {A-T} and G can bind with its complementary C {G-C} for DNA, whereas {A-U} and {G-C} for RNA. This is the basic principle of a microarray.

The different types of available microarrays can be classified in the following five classes:

– *printed microarrays*. Printed microarrays can be classified as a contact or noncontact printing. The noncontact printer blows the biological unit droplets onto the microarray's surface by using a technology similar to that used in a computer printer (e.g., ink-jet). In contact printing, the print head hits the surface, applying the probe solution directly onto the microarray's surface. In both cases, only a few quantities of probe solution are used per single spot. Another distinctive element that allows to classify printed microarrays is the nature of probes, the double-stranded DNA (dsDNA) [5] and the oligonucleotide [6]. dsDNA microarray probes are obtained by using the polymerase chain reaction (PCR) and are called amplicons.

Sequences length range from a few hundred bases to a thousand. In dsDNA, genes are identified each 200–800 bases pair (bp). dsDNA presents high sensitivity and low specificity that can be improved by using redundancy methodologies of sequences into probes. In oligonucleotide microarrays, probes are obtained from a chemical synthesized sequence. In oligonucleotide microarrays, the length of probes ranges from 25 to 80 bp, reaching 150 bp for gene expression microarrays. Shorter probe length allows to reduce the number of errors during the synthesis phase as well as simplify the interrogation of small genomic regions including the polymorphisms. On the other hand, shorter probe length may adversely affect sensitivity compared to dsDNA probes; whereas specificity is often higher when short, that is, interrogating specific genomic regions.

- In situ-synthesized oligonucleotide microarrays (iSSOM) are known as very high-density microarrays and use oligonucleotide probes. The most known iSSOM is the GeneChip produced by Affymetrix [7, 8]. In iSSOM, probes are synthesized directly on the surface of the microarray typically made of glass or quartz. Probes are synthesized using semiconductor-based photochemical synthesis. The average probe length in iSSOM ranges between 20 and 25 bp; to improve statistical accuracy, specificity, and sensitivity multiple probes per target are added. Other high-density oligonucleotide microarrays include those manufactured by Roche NimbleGen and Agilent Technologies. NimbleGen uses maskless photo-mediated synthesis, and Agilent employs inkjet technology for in situ manufacturing of probes. The expense of a custom Affymetrix microarray is high and expensive to be customized for diagnostic purposes; in contrast, NimbleGen and Agilent can be easily customized for diagnostic purpose.
- High-density bead microarrays (HDBM) [9, 10]. HDBM technology is produced by Illumina, making HDBM appropriate for whole-genome genotyping, copy number variation (CNV) detection. Because the beads in BeadArrays randomly assort to their final location on the array, the bead location must be mapped. The mapping of the Illumina beads is achieved by a group of hybridization and rinse methods, enabling fluorescently labeled complementary oligonucleotides to bind to their specific bead sequences. HDBM have been successfully applied to and SNP genotyping, including the International HapMap Project.
- *Electronic microarrays* (EM). In EM nucleic acid move through the electric fields, thus the hybridization process is controlled by electric fields. Nanogen is the developer of electronic technology that allows for transport and hybridization of DNA on a semiconductor chip [11, 12]. EMs rely on semiconductor-

connectors that control each test sites (connectors and test sites depend on the chosen microarray). Electronic microarrays allow fast and accurate delivery of electronically charged biological molecules to test sites (electrodes) on microarrays. Negatively charged nucleic acids are transported to specific locations when a positive current flow goes through one or more spot on the microarray. At the end of hybridization, the hybridized probes can be read through a scanner, to detect the red and green colors obtained by applying labels fluorescently (normally an isotope).

– Suspension bead microarrays (SBA) are three-dimensional microarrays described for the first time in 1977 in [13]. SBAs use microsphere beads (56 μm in diameter), allowing for the concurrent testing of multiple gene variants. Each microsphere bead has a unique identifier relying on variations in optical features, that is, in fluorescent color. The intensity defines the color of each bead easily discriminated through their wavelength intensity. Similar to flat microarrays (e.g., DNA microarray), hybridization happens between a suitable receptor and a probe that attaches itself to different microspheres [14, 15]. Probe–target hybridization is detected by optical targets, determining the relative abundance of each target in the sample.

Compared with iSSOMs, PMs are relatively inexpensive and straightforward. The significant benefits of printed microarrays are flexibility and the capability to study organisms that have not been fully sequenced. As a drawback, there is difficulty in employing PMs and iSSOMs in clinical diagnostics.

Unlike the known locations of printed and in situ-hybridized microarrays, the beads in HDBMs randomly assort to their final location on the array. Thus, the bead location needs to be mapped, through an opportune decoding process.

The PMs, iSSOMs, and HDBMs rely on passive transport of nucleic acids for the hybridization process. Conversely, EMs employ active hybridization through electric fields to control the transport of nucleic acids. Moreover, EMs are a more flexible technology and is more practical for diagnostic applications.

SBAs is the most attractive platform for high-throughput nucleic acid detection in clinical diagnostics.

## 3 Software Tools to Analyze Microarray Data

In the last decade, microarrays have become one of the most valuable methodologies in genomic research, due to the considerable amount of data produced per single experiment. The microarray outcomes involve the analysis of large datasets of information derived from each single biological experiment. Feature that made

microarrays one of the most valuable methodologies in genomic research, due to the considerable amount of data produced per single experiment, making possible to study problem from a broader and better prospective especially complex diseases such as cancer. Microarrays data analysis depend on the type of employed microarray. Gene expression analysis involves the monitoring of the expression levels of thousands of genes simultaneously, it is necessary use specific tools able to deal with numerical data. Conversely to analyze data obtained by using oligonucleotides microarrays, it is mandatory to use different tools form that employed in gene expression analysis, because, in this case it is necessary to handle textual data.

Elucidating the statistical relevance of these huge amount of data obtained from microarray experiments is a challenge. Dealing with huge datasets requires skillful statistical and computational support, since manually data analysis is infeasible as well as error-prone task. Thus, the development of software tools that can handle these large datasets is mandatory, in order to simplify and speed up the work of biologists. Moreover, the number of applications where microarrays are involved is increasing faster than the development of suitable data analysis models and tools, generating an increasing demand for adequate statistical and data mining tools. On the other hand, the necessity to provide researchers with scalable and easy-to-use software tools arises as well. Thus, to extract biologically relevant information contained in microarray datasets, we report a list of available tools with which we could analyze microarray datasets. In this section we try to highlight their main features.

- Automated Microarray Data Analysis (ADMA) [16] is freely available for download as an R [17] package under the GPL license, at the following web address: [18]. AMDA is written by using the R language [19]. The R/Bioconductor project [20] has become the reference open-source software project for the analysis of Affymetrix [21] microarray datasets, including normalization, election of differentially expressed genes, image analysis, quality controls, and clustering. Moreover, R/Bioconductor is easy to install on the most common operating systems (Linux, Mac OS X, Windows), making ADMA available with all the major operating systems. AMDA is available as a stand-alone R tool; help is available in the standard R documentation format.
- Microarray Data Analysis System (MIDAS) [22–24] is a software application written in Java, to preprocess Affymetrix data. The main advantage of the software written in Java [25] is the compatibility with all the most known and used operating systems such as Unix/Linux, Mac OS, and Windows. The only requirement for the user to be possible to use MIDAS is that, users have already installed Java on his/her machine. MIDAS

is available freely for the download at the following web site: https://sourceforge.net/projects/midas-tm4/files/latest/download?source=files. It is made available under freeware license for any academic or other noncommercial purposes. To execute MIDAS on his/her machine users should use the provided ".sh" file on Linux and Mac OS machines or the ".bat" file on Windows machines. MIDAS comes with an intuitive graphical user interface (GUI) with which to design analysis workflow combining one or more normalization and filtering functions. Moreover, the results of the analysis workflows can be conveyed in graphical format such as scatterplots, and box-plots making easier for the users to investigate hidden trend into the data. MIDAS can deal with CDF (channel file) and CEL (intensity file) files. At the end of the analysis workflow, MIDAS allows user to export data in "tav" file format, or export report in pdf or plain-text files.

- Meta-Analysis of Affymetrix Microarray Data analysis (MAAMD) [26–31] is an automated tool developed in Kepler to create analysis workflow with which to execute data downloading, data organization, data quality control, pathway visualization, differential gene expression analysis, clustering analysis, gene-set enrichment analysis, and cross-species orthologous-gene comparisons. MAAMD is freely available as stand-alone software tools, for R, AltAnalyze, and as Bioconductor packages [20]. Moreover, GEOquery and arrayQualityMetrics modules are embedded in MAAMD. The inputs of MAAMD are CEL and csv files, which contain sample information and parameters describing the locations of input files and required tools. Moreover, input files are user-editable. Outcomes of MAAMD are arranged in a structured folder containing the microarray data and analyzed results. Users in order to use MAAMD should have preinstalled on his/her machine Kepler 2.4 or above [32], AltAnalyze 2.0.8 or above [33], JDK 7 or above [25], and R 3.0.0 or above [17]. MAAMD is compatible with Windows and Mac OSX operating systems, and it is freely available for academic or noncommercial use at the following address: https://www.biokepler.org/use_cases/maamd-workflow-standardize-meta-analyses-affymetrix-microarray-data.
- GSEA [34, 35] is a software implemented in Java and R languages. The desktop version GSEA comes with and easy-to-use graphical interface through run the analysis workflows. In addition, GSEA is available as Java jar file providing only a Command line interface that may be useful for analyzing several datasets sequentially, analyzing large datasets, or running analyses on a compute cluster. R-GSEA, is an R implementation of GSEA. GSEA supports input data as default ASCII tab-delimited text files. Other different text files are identified by means of special

extensions. Extensions available are .gct to identify expression data, ".cls" for phenotypes, ".gmt" for gene sets, and .chip for chip annotations. After loaded the data files, it is possible to run the gene set enrichment analysis. GSEA is available for download at http://software.broadinstitute.org/gsea/login.jsp under BSD3-clause "New" or "Revised" license, after full filled the registration form. GSEA is available for MacOs and Linux operating systems.

- CisGenome is a bioinformatics platform to create ChIP-seq analysis workflows [36, 37]. CisGenome is available for Linux/Unix, Windows, and Mac OS operating systems. It can be freely downloaded for academic or noncommercial uses at http://www.biostat.jhsph.edu/~hji/cisgenome/index_files/download.htm. CisGenome is entirely written by using C [38] and C++ [39] programming languages. CisGenome is available as command line tools (core programs only) version suitable for analyzing large amounts of data effectively on cluster computers. CisGenome requires advanced computer science skills to be used. Desktop version comes with a graphical user interface providing the same functionalities of command line version, but it is easier to use. CisGenome allows users to perform data normalization, data visualization, false discovery rate (FDR) computation, peak detection, gene-peak association, sequence analysis, and motif analysis.
- Birdsuite [40, 41] is an open-source collection of tools, with which to discover SNP genotypes, common copy-number polymorphisms (CNPs), and de novo CNVs in samples obtained by using Affymetrix platform. The current version of Birdsuite is available only for Linux operating system. To make Birdsuite compatible with other operating systems, it is necessary to download and use the Matlab version. Both versions of Birdsuite can be downloaded at https://www.broadinstitute.org/birdsuite/birdsuite-downloads.

  To run Birdsuite users have to install the following components: Java 1.5. or higher [25], Python 2.5.2 or higher [42], numpy Python package, version 1.2 or higher [43], R 2.4 or higher [17], mclust R package, version 3 or higher [44], Affymetrix Power Tools (APT) 1.8.6 or 1.10.2, binaries [45] and Affymetrix library bundle for Genome-Wide Human SNPArray6.0. Birdsuite is specially designed for integrated analysis of SNPs and CNVs.
- EXpression Analyzer and DisplayER (EXPANDER) [46–48] is a framework for the creation of gene expression data analysis workflows. EXPANDER is written in Java characteristic that make it compatible with all operating systems compatible with the Java technology. Expander can be freely downloaded only for academic or noncommercial use. The preprocessing of CEL file and the SAM filter utility require the preinstallation of R, for

statistical computing and graphics. Files with suffix BED and GFF3 are supported as input in the ChIP-Seq file format, while Gene Ranking Analysis files supported as input present the GSEA suffix; for the other kinds of files there is no suffix limitation. EXPANDER provides data preprocessing, Clustering and Biclustering GE Data analysis and visualization, Network Based Grouping of GE Data, integrative analysis of ChIP-Seq and Gene Expression Data, General and specific Enrichment Analysis, Network Based Enrichment Analysis, Matrix Visualizations, and PCA Transformation.

- lumi data preprocessing of Illumina [49] bead microarray analysis. lumi [50] is a framework developed by Illumina for bead microarray to perform methylation and expression analysis [51, 52]. lumi is freely available for download at http://bioconductor.org/packages/release/bioc/html/lumi.html under GNU Lesser General Public License version 3.0. It comes with a command line interface, and it is written in R language; thus preinstalling R on the computer where lumi will be used is a prerequisite. lumi is compatible with Unix/Linux, Mac OS, and Windows operating systems.
- TreeView gene expression visualization framework [53]. TreeView is developed in Java and is freely available for download at https://sourceforge.net/projects/jtreeview/files/ under the GNU General Public License version 2.0. It compatible with Unix/Linux, Mac OS, and Windows operating systems, requiring the preinstallation of Java to work properly. TreeView comes with a simple graphical user interface with which user can visualize and analyze microarray data. At the end of data analysis phase, results are conveyed by means dendrogram, scatterplot, and karyoscope charts, providing visual clues on the principal genes in a study. TreeView allows to export results in image formats such as PNG, PPM, and JPEG; it also supports export to vector-based postcript files. Additionally, it is possible to export subsets of results as a tab-delimited text, that is, gene lists and cdts. Command line interface is very useful to analyze huge datasets by taking advantage of high-performance computers (i.e., clusters). Command line provides all the features of GUI version with the only difference that command should be typed as argument of the Java virtual machine.
- DMET-Analyzer [54] is a tool for the automatic association analysis of variation of patient genomes and clinical conditions of patients, that is, different responses to drugs. DMET-Analyzer allows for automatizing the workflow of analysis of DMET datasets, avoiding the use of multiple tools, as well as the automatic annotation of SNP data retrieval information in the existing databases of SNPs (e.g., dbSNP), and the association of SNP within a pathway retrieving information in

specialized databases (e.g., PharmaGKB). DMET-Analyzer input files have to be in xlsx or plain tab-delimited format. All the results as well as annotated result can be easily saved by clicking on it and saving it in txt format, html, or so on. DMET-Analyzer is written in Java; it presents a simple graphical user interface that allows users (doctors/biologists) to analyze DMET files interactively produced by Affymetrix DMET-Console. Moreover, FDR and Bonferroni statistical corrector are available, as well as the Hardy–Weinberg equilibrium calculator. DMET-Analyzer is written in Java and is available for Unix/Linux, Mac OS, and Windows operating systems. DMET-Analyzer is freely available under the GNU General Public License version 2.0 for download at the following address https://sourceforge.net/projects/dmetanalyzer/.

- *PARES* (Parallel Association Rules Extractor from SNPs) [55] is a software tool developed in Java for the parallel extraction of association rules from datasets obtained by using DMET microarrays. PARES can handle input files in a tabular plain text format or in xls (Excel) format. Result can be exported in txt file or rtf (rich text format). Associative rules mined by PARES can correlate the presence of a set of allelic variants with the clinical condition of patients. PARES thanks to the simple and intuitive graphic user interface is a software tool that allows to easily extract multiple relations between genomic factors buried in the datasets. PARES is compatible with Windows, Linux/UNIX, and Mac OS operating systems. It is distributed under Creative Commons license and is available for free download for academic and not-for-profit institutions, at the following web address https://sites.google.com/site/pareswebsite/pares.
- *OS-Analyzer (OSA)* [56] is a software tool written in Java for the computation and visualization of Overall Survival (OS) and Progression Free Survival (PFS) curves of cancer patients and evaluate their association with ADME gene variants. OS-Analyzer can perform automatic analysis of DMET datasets enriched with survival events. OS-Analyzer can handle DMET data annotated with temporal information in tabular plain text format or in xls (Excel) file format. Results can be exported as image file (png, tiff) by clicking on the survival chart, together with overall survival information. It is distributed under Creative Commons license and is available for free download for academic and not-for-profit institutions, at the following web address: https://sites.google.com/site/overallsurvivalanalyzer/os-analyzer.
- *DMET-Miner* [57] is a software tool for the automatic mining of association rules, correlating the presence of a group of allelic variants with the clinical condition of subjects, (e.g., the combination of alleles typical of a class responsible for the different response to drugs). DMET-Miner allows users to calculate

automatically and directly association rules from a whole DMET (Drug Metabolism Enzymes and Transporters) dataset. DMET-Miner can handle input file in tabular plain text format or in xls (excel) format. DMET-Miner is written in Java, and it presents a simple graphical user interface, allowing the users to analyze a dataset through some mouse clicks. DMET-Miner is distributed under Creative Commons license and is available for free download for academic and not-for-profit institutions, at the following web address https://sites.google.com/site/dmetminer/get-software.

– cloud4SNP [58] is a Cloud-based bioinformatics tool for the parallel preprocessing and statistical analysis of pharmacogenomics SNP DMET microarray data. It is a Cloud-based version of *DMET-Analyzer* [54], that has been implemented on the Cloud using the Data Mining Cloud Framework [59], a software environment for the design and execution of knowledge discovery workflows on the Cloud [60]. It allows to statistically test the significance of the presence of SNPs in two classes of samples using the well-known Fisher test. *Cloud4SNP* uses data parallelism and employs an optimized methodology to avoid the execution of useless Fisher tests, through filtering of probes with similar SNPs distributions.

## 4 Discussion

Tools for microarray data analysis give an opportunity to generate functional data on a genome-wide scale and, consequently, should provide much needed data for the biological interpretation of genes and their functions. To pursue this goal, the choice of the most suitable analysis tool is mandatory, taking into account the plethora of available software tools. The several tools make the analysis, handling, and understanding of microarray data a vague science. For these reasons, a basic knowledge of the employed computational tools is therefore required for optimal experimental design and meaningful data analysis, to extract biological relevant information contained in microarray datasets.

To help researchers to choose the most suitable tool for his/her purpose in Table 1 are summarized the main features provided of each listed tool in the previous section.

The tools listed above are thought to help the researcher to perform microarray data analysis in an easy and fast way. Each tool is developed to be easy to use without losing accuracy in the provided results.

**Table 1**
**The table summarizes the main features provided by each listed tool**

| ToolName | OS | Lic | InFile | OFile | GUI | Vis | DM | SA | CNV | GE | DE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ADMA | WLM | OSource | CEL | txt, images | × | √ | × | × | × | √ | × |
| MIDAS | WLM | FW | CDF, CEL | tav, pdf, txt, images | √ | √ | × | × | × | × | × |
| MAAMD | WM | FAnC | CEL, csv | folder | √ | √ | × | × | × | √ | √ |
| GSEA | LM | BSD3 | txt | txt, images | √ | √ | × | × | × | √ | × |
| CisGenome | WLM | FAnC | txt | txt, images | √ | √ | × | √ | × | √ | × |
| Birdsuite | L | OSource | txt | txt, images | √ | √ | × | √ | √ | × | √ |
| EXPANDER | WLM | FAnC | BED, GF3 | txt, images | √ | √ | × | × | × | √ | √ |
| lumi | WLM | GnuL | txt | txt, images | √ | √ | × | × | × | √ | × |
| TreeView | WLM | GnuL2 | txt | images, ps | √ | √ | × | × | √ | × | × |
| DMET-Analyzer | WLM | FAnC | txt, xlsx | txt, images | √ | √ | × | √ | × | × | √ |
| PARES | WLM | FAnC | txt, xlsx | txt, images | √ | √ | √ | × | × | × | × |
| OS-Analyzer | WLM | FAnC | txt, xlsx | txt, images | √ | √ | × | √ | × | × | √ |
| DMET-Miner | WLM | FAnC | txt, xlsx | txt, images | √ | √ | √ | × | × | × | √ |
| cloud4SNP | WLM | FAnC | txt, xlsx | txt, images | √ | √ | × | √ | × | × | × |

In the table, OS refers to operating system, Lic is the abbreviation of license, InFile stands for input file, and OFile stands for output file. Vis refers to Visualization, DM refers to Data Mining, SA refers to statistical analysis, CNV refers to of copy number variation, GE refers to of Gene expression analysis, and DE refers to Data Enrichment. WLM refers to Windows, Linux, and MacOS, FW refers to of Free-Ware, OSource refers to Open Source, FAnC stands for Free for academic and not-for-profit institutions

## 5 Conclusion

Microarrays allow for the investigation of genetic variations underlying interindividual variabilities in drug pharmacokinetics/pharmacodynamics, providing a complete understanding of gene function, regulation, and interactions. However, to exploit all the power of this massive amount of data in the shortest time possible, the necessity to develop software tools for efficient data collection and analysis arises. The development of efficient and scalable software tools avoids researchers being overwhelmed with the ever-growing flow of data. Thus, tools that are able to deal with these enormous amounts of data as well as easy to use can speed up the process of data analysis and knowledge extraction. In this chapter, we review some software tools available in the literature for dealing with microarray data. The abundance of software tools is due to the board spectrum of microarray analysis methods along with the massive amount of data produced by a single microarray

experiment, which makes impossible for a single tool to highlight remarkably all the clues hidden in the data. Thus, the necessity of multiple software tools that are able to examine accurately a single aspect of the problem under investigation is essential.

## References


1. Watson JD, Crick FH (1953) Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. Nature 171:737–738
2. Sanger F, Nicklen S, Coulson AR (1977) DNA sequencing with chain-terminating inhibitors. Proc Natl Acad Sci U S A 74(12):5463
3. Schildkraut CL, Marmur J, Doty P (1961) The formation of hybrid DNA molecules and their use in studies of DNA homologies. J Mol Biol 3(5):595
4. Grunstein M, Hogness DS (1975) Colony hybridization: a method for the isolation of cloned DNAs that contain a specific gene. Proc Natl Acad Sci U S A 72(10):3961
5. Wang J, Bai Y, Li T, Lu Z (2003) DNA microarrays with unimolecular hair-pin double-stranded DNA probes: fabrication and exploration of sequence-specific DNA/protein interactions. J Biochem Biophys Methods 55 (3):215
6. Sussman MJ (1999) i, United States patent, Nature biotechnology 17, 974
7. Gautier L, Cope L, Bolstad BM, Irizarry RA (2004) affy—analysis of Affymetrix GeneChip data at the probe level. Bioinformatics 20 (3):307
8. Irizarry RA, Bolstad BM, Collin F, Cope LM, Hobbs P, Speed TP (2003) Summaries of Affymetrix GeneChip probe level data. Nucleic Acids Res 31(4):e15
9. Fan JB, Gunderson KL, Bibikova M, Yeakley JM, Chen J, Garcia EW, Lebruska LL, Laurent M, Shen R, Barker D (2006) Illumina universal bead arrays. Methods Enzymol 410:57
10. Shen R, Fan JB, Campbell D, Chang W, Chen J, Doucet D, Yeakley J, Bibikova M, Garcia EW, McBride C et al (2005) High-throughput snp genotyping on universal bead arrays. Mutat Res 573(1):70
11. Blin A, Cisse I, Bockelmann U (2014) Electronic hybridization detection in microarray format and DNA genotyping. Sci Rep 4:4194
12. Sosnowski RG, Tu E, Butler WF, O'Connell JP, Heller MJ (1997) Rapid determination of single base mismatch mutations in DNA hybrids by direct electric field control. Proc Natl Acad Sci 94(4):1119
13. Horan PK, Wheeless LL (1977) Quantitative single cell analysis and sorting. Science 198 (4313):149
14. Schwenk JM, Gry M, Rimini R, Uhl'en M, Nilsson P (2008) Antibody suspension bead arrays within serum proteomics. J Proteome Res 7(8):3168
15. Gleason LU, Burton RS (2012) High-throughput molecular identification of fish eggs using multiplex suspension bead arrays. Mol Ecol Resour 12(1):57
16. Pelizzola M, Pavelka N, Foti M, Ricciardi-Castagnoli P (2006) AMDA: an R package for the automated microarray data analysis. BMC Bioinformatics 7(1):335
17. http://cran.r-project.org/
18. https://sourceforge.net/projects/automicroarray/files/latest/download?source=typ_redirect
19. Ihaka R, Gentleman R (1996) R: a language for data analysis and graphics. J Comput Graph Stat 5(3):299
20. Gentleman RC, Carey VJ, Bates DM, Bolstad B, Dettling M, Dudoit S, Ellis B, Gautier L, Ge Y, Gentry J, Hornik K, Hothorn T, Huber W, Iacus S, Irizarry R, Leisch F, Li C, Maechler M, Rossini AJ, Sawitzki G, Smith C, Smyth F, Tierney L, Yang JY, Zhang J (2004) Bioconductor: open software development for computational biology and bioinformatics. Genome Biol 5(10): R80
21. https://www.thermofisher.com/us/en/home/life-science/microarray-analysis.html
22. Saeed A, Sharov V, White J, Li J, Liang W, Bhagabati N, Braisted J, Klapa M, Currier T, Thiagarajan M et al (2003) Tm4: a free, open-source system for microarray data management and analysis. BioTechniques 34(2):374
23. Dudoit S, Gentleman RC, Quackenbush J et al (2003) Open source software for the analysis of microarray data. BioTechniques 34(13):45
24. Saeed AI, Bhagabati NK, Braisted JC, Liang W, Sharov V, Howe EA, Li J, Thiagarajan M, White JA, Quackenbush J (2006) tm4 microarray software suite. Methods Enzymol 411:134

25. http://www.oracle.com/technetwork/java/javase/downloads/index.html
26. Gan Z, Wang J, Salomonis N, Stowe JC, Haddad GG, McCulloch AD, Altintas I, Zambon AC (2014) MAAMD: a workflow to standardize meta-analyses and comparison of Affymetrix microarray data. BMC Bioinformatics 15 (1):69
27. Altintas I, Wang J, Crawl D, Li W (2012) Workshop on Data analytics in the Cloud (DanaC2012) at EDBT/ICDT 2012 conference. http://www.edbt.org/Proceedings/2012-Berlin/papers/workshops/danac2012/a5-altintas.pdf
28. Wang J, Crawl D, Altintas I (2012) 1st international workshop on advances in the Kepler scientific workflow system and its applications at ICCS 2012 conference
29. Wang J, Altintas I (2012) 1st international workshop on advances in the Kepler scientific workflow system and its applications at ICCS 2012 conference
30. Crawl D, Wang J, Altintas I (2011) Provenance for MapReduce-based data-intensive workflows. In: Supercomputing 2011 (SC2011) conference, ACM 2011, vol Proceedings of the 6th workshop on workflows in support of large-scale science (WORKS11). http://users.sdsc.edu/~jianwu/JianwuWang_files/Provenance_for_MapReduce-based_Data-Intensive_Workflows-2011.pdf
31. Altintas I (2011) Distributed workflow-driven analysis of large-scale biological data using biokepler. In: Proceedings of the 2nd international workshop on Petascal data analytics: challenges and opportunities (ACM, New York, NY, USA). Doi:https://doi.org/10.1145/2110205.2110215
32. https://kepler-project.org/users/downloads
33. http://www.altanalyze.org/
34. Mootha VK, Lindgren CM, Eriksson KF, Subramanian A, Sihag S, Lehar J, Puigserver P, Carlsson E, Ridderstråle M, Laurila E, Houstis N, Daly MJ, Patterson N, Mesirov JP, Golub TR, Tamayo P, Spiegelman B, Lander ES, Hirschhorn JN, Altshuler D, Groop LC (2003) PGC-1alpha-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes. Nat Genet 34:267–273
35. Subramanian A, Tamayo P, Mootha VK, Mukherjee S, Ebert BL, Gillette MA, Paulovich A, Pomeroy SL, Golub TR, Lander ES, Mesirov JP (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proc Natl Acad Sci U S A 102(43):15545
36. Ji H, Jiang H, Ma W, Johnson DS, Myers RM, Wong WH (2008) An integrated software system for analyzing chip-chip and chip-seq data. Nat Biotechnol 26:1293. EP
37. Zhou Q, Wong WH (2004) CisModule: de novo discovery of cis-regulatory modules by hierarchical mixture modeling. Proc Natl Acad Sci U S A 101(33):12114
38. http://www.open-std.org/jtc1/sc22/wg14/
39. http://www.open-std.org/jtc1/sc22/wg21/
40. Korn JM, Kuruvilla FG, Mc-Carroll SA, Wysoker A, Nemesh J, Cawley S, Hubbell E, Veitch J, Collins PJ, Darvishi K, Lee C, Nizzari MM, Gabriel SB, Purcell S, Daly MJ, Altshuler D (2008) Integrated genotype calling and association analysis of snps, common copy number polymorphisms and rare cnvs. Nat Genet 40:1253–1260
41. McCarroll SA, Kuruvilla FG, Korn JM, Cawley S, Nemesh J, Wysoker A, Shapero MH, de Bakker PIW, Maller JB, Kirby A, Elliott AL, Parkin M, Hubbell E, Webster T, Mei R, Veitch J, Collins PJ, Handsaker R, Lincoln S, Nizzari M, Blume J, Jones KW, Rava R, Daly MJ, Gabriel SB, Altshuler D (2008) Integrated detection and population-genetic analysis of snps and copy number variation. Nat Genet 40:1166. EP
42. http://www.python.org
43. http://numpy.scipy.org
44. http://lib.stat.cmu.edu/R/CRAN/web/packages/mclust/index.html
45. http://www.affymetrix.com/support/developer/powertools/index.affx
46. Sharan R, Maron-Katz A, Shamir R (2003) Click and expander: a system for clustering and visualizing gene expression data. Bioinformatics 19(14):1787
47. Ulitsky I, Maron-Katz A, Shavit S, Sagir D, Linhart C, Elkon R, Tanay A, Sharan R, Shiloh Y, Shamir R (2010) Expander: from expression microarrays to networks and functions. Nat Protoc 5:303. EP
48. Shamir R, Maron-Katz A, Tanay A, Linhart C, Steinfeld I, Sharan R, Shiloh Y, Elkon R (2005) Expander z an integrative program suite for microarray data analysis. BMC Bioinformatics 6(1):232
49. https://emea.illumina.com/
50. Du P, Kibbe WA, Lin SM (2008) lumi: a pipeline for processing illumina microarray. Bioinformatics 24(13):1547
51. Du P, Zhang X, Huang CC, Jafari N, Kibbe WA, Hou L, Lin SM (2010) Comparison of beta-value and m-value methods for

quantifying methylation levels by microarray analysis. BMC bioinformatics 11(1):587

52. Lin SM, Du P, Huber W, Kibbe WA (2008) Model-based variance-stabilizing transformation for Illumina microarray data. Nucleic Acids Res 36(2):e11
53. Saldanha AJ (2004) Java Treeview—extensible visualization of microarray data. Bioinformatics 20(17):3246
54. Guzzi PH, Agapito G, Di Martino MT, Arbitrio M, Tassone P, Tagliaferri P, Cannataro M (2012) DMET-Analyzer: automatic analysis of Affymetrix DMET data. BMC Bioinformatics 13(1):258
55. Agapito G, Guzzi PH, Cannataro M (2019) Parallel extraction of association rules from genomics data. Appl Math Comput 350:434–446
56. Agapito G, Botta C, Guzzi PH, Arbitrio M, Di Martino MT, Tassone P, Tagliaferri P, Cannataro M (2016) OSAnalyzer: a bioinformatics tool for the analysis of gene polymorphisms en-riched with clinical outcomes. Microarrays 5(4). https://doi.org/10.3390/microarrays5040024
57. Agapito G, Guzzi PH, Cannataro M (2015) DMET-Miner: Efficient discovery of association rules from pharmacogenomic data. J Biomed Inform 56:273
58. Agapito G, Cannataro M, Guzzi PH, Marozzo F, Talia D, Trunfio P (2013) Proceedings of the international conference on bioinformatics, computational biology and biomedical informatics
59. Marozzo F, Talia D, Trunfio P (2013) A cloud framework for big data analytics workflows on azure. In: Grandinetti L (ed) Clouds, grids and big data. IOS Press, Chap. Big Data
60. Marozzo F, Talia D, Trunfio P (2012) European conference on parallel processing. Springer, pp 220–227

# Chapter 14

## Challenges and Future Trends for Microarray Analysis

**Verónica Bolón-Canedo, Amparo Alonso-Betanzos, Ignacio López-de-Ullibarri, and Ricardo Cao**

### Abstract

The current situation in microarray data analysis and prospects for the future are briefly discussed in this chapter, in which the competition between microarray technologies and high-throughput technologies is considered under a data analysis view. The up-to-date limitations of DNA microarrays are important to forecast challenges and future trends in microarray data analysis; these include data analysis techniques associated with an increasing sample sizes, new feature selection methods, deep learning techniques, covariate significance testing as well as false discovery rate methods, among other procedures for a better interpretability of the results.

**Key words** Big data, Deep learning, False discovery rate, Feature selection, Microarray data, Significance testing

## 1 Introduction

In the last decade DNA microarray technologies have been coexisting with new competitive high-throughput methodologies based on sequencing. In general, studies designed for techniques, for example, gene-expression arrays, ChIP-on-chip (a technology that combines chromatin immunoprecipitation—"ChIP"—with DNA microarray chip), or SNP arrays (single nucleotide polymorphism, a type of DNA microarray which is used to detect polymorphisms within a population), could equally have been conducted by using next-generation sequencing (NGS) procedures like, respectively, RNA-seq, ChIP-seq, or genotyping-by-sequencing. Given the much broader biological and technical insight of NGS, all experts have agreed in predicting a progressive abandon of arrays in favor of NGS methods (see, e.g., [1, 2]). A quick search on Medline along the last years can convince the reader of the plausibility of this prediction.

Currently, it appears that the main factor that could still induce researchers to opt for array technologies in detriment of newer

Verónica Bolón-Canedo and Amparo Alonso-Betanzos (eds.), *Microarray Bioinformatics*, Methods in Molecular Biology, vol. 1986, https://doi.org/10.1007/978-1-4939-9442-7_14, 

NGS alternatives is the better cost effectiveness of the former. In this sense, although an objective, comprehensive evaluation of price evolution would be difficult, time is on the side of NGS, since available evidence indicates that the gap between the two technologies is progressively narrowing (see, e.g., the graphs at https://www.genome.gov/27541954/dna-sequencing-costs-data/). Of course, given the inherent constraints of hybridization technologies, it must be stressed that the cost advantage of microarrays cannot be exploited, e.g., in studies aimed at gene discovery or involving poorly characterized organisms, because then the probes simply will not exist. On the other hand, in the case of large-sample studies NGS will probably continue to be unaffordable for a long time [2]. Besides, realistic measures of cost should consider not only the market price of the technological product, but also other components like workflow complexity and availability of data-analytic resources. These aspects may add to the advantages of microarrays over NGS. For example, some issues related to the choice of processing workflow of (bulk and single-cell) RNA-seq data are not yet settled [3–5], which conditions the interpretation of results.

All in all, and for the sake of concreteness focusing on gene-expression analysis, RNA-seq has become (or is becoming) the gold standard in general transcriptome studies [5, 6]. To date, the only possible exception where microarrays would outperform RNA-seq would be found in studies aimed at the characterization of the isoforms produced by alternative splicing of the exons. In that setting, Nazarov et al. [7] conclude that, for the usual sequencing depths, RNA-seq would suffer from reproducibility problems concerning long non-coding RNAs. Git et al. [8] had previously pointed out that, with the platforms they compared, RNA-seq performed worse than microarrays in detecting differential microRNA expression.

Perhaps, one could imagine a niche for the use of microarrays in clinical applications of genetic testing [9, 10], where the list of genes involved is typically well defined and the plethora of information produced by NGS would be excessive. On the other hand, it is true that for these low- to medium-throughput applications quantitative PCR (qPCR) is also available, besides having a reputation of being a "gold standard." Even more, newer hybridization-based technologies like the Nanostring nCounter analysis system have also been shown to perform promisingly in these settings [9, 11].

In the field of proteomics, the competing relationship between protein microarrays and mass spectrometry may to some degree resemble that between DNA microarrays and NGS, as described above. According to some authors (*see* [12]) it would be more accurate to perceive that relationship as one of the complementariness. Currently, protein microarray technology is still a developing area [13]. From a data-analytic viewpoint, general procedures

developed for DNA microarrays in relation with, e.g., experimental design, preprocessing, and differential expression analysis are, in principle, extensible to protein microarrays.

The content of the rest of the chapter is as follows. Subheading 2 includes a view of the current limitations of DNA microarray data analysis. The challenges and future trends foreseen by the authors for the near future are included in Subheading 3. They include data analysis techniques related to the increasing sample size issue, new feature selection methods, deep learning techniques as well as interpretation statistical tools, as covariance significance testing and false discovery rate methods. Finally, some conclusions on the subjects discussed are briefly outlined in Subheading 4.

## 2 Limitations of DNA Microarray Data Analysis

The use of microarrays provides a powerful technology in the field of genetics. They have revolutionized the concept of patient-tailored treatment since they allow in-depth analysis of gene-expression profiles. Before microarrays being fully qualified as a useful clinical tool they must demonstrate reliability and reproducibility. The nature of microarray experiments imposes a large number of limitations. These apply to simple issues as sample acquisition and data mining, as well as to more controversial ones related to the data analysis methods required for the enormous amount of information available. Methods for validating gene-expression profiles and those for improving trial designs are among the priorities for the near future. On the other hand, in many setups microarrays are being rapidly replaced by DNA sequencing technologies.

Microarrays are just devices to measure, in a simultaneous way, the relative concentrations of different DNA or RNA sequences. They have been very useful in many applications, but some of their limitations will be considered in this section. Arrays are used to construct an indirect measure of relative concentration. Thus the value measured at a given position on a microarray is often assumed to be proportional to the concentration of a species that can hybridize to that location. However, the level at a given location on the array is not fully proportional to concentration of the species hybridizing to the array. At high concentrations the array becomes saturated and at low concentrations no binding is very common. As a consequence, the measured quantity is linear only on a range of concentrations. On the other hand, it is often difficult to design arrays in which multiple related DNA/RNA sequences will not bind to the same probe on the array. A sequence designed to detect a specific gene may also detect other genes, if those ones have significant sequence homology to the first gene. This can be particularly problematic for gene families and for genes with multiple splice variants. On the other hand, arrays can be designed to

specifically detect splice variants. However, it is difficult to design arrays that will uniquely detect every exon or gene in genomes with multiple related genes.

It is important to stress that microarrays can only detect sequences that they were designed to detect. Thus, if the solution being hybridized to the array contains RNA or DNA species for which there is no complimentary sequence on the array, those species will not be detected. This means that genes not yet been annotated in a genome will not be represented on the array. In addition, non-coding RNAs that are not yet recognized as expressed are typically not represented on an array. For highly variable genomes, arrays are typically designed using information from the genome of a reference strain and a large fraction of the genes present may be missing. Consequently an array designed using gene annotation may not contain many important genes.

## 3 Challenges and Future Trends for DNA Microarray Analysis

### 3.1 Increasing the Number of Samples

A typical characteristic of microarray datasets is their small sample size (usually less than 100 examples). This is because, especially in the beginning, the sequencing cost was high, preventing its application to a significant number of individuals. The small sample size has been a major issue for data analysts, since it makes very difficult to correctly evaluate their models.

It was back in 2001 when Dougherty [14] noted that the error estimation is greatly impacted by small samples in microarray data classification. Without the appropriate estimation of the error, an unsound application of classification methods follows, which has generated a large number of publications and an equally large amount of unsubstantiated scientific hypotheses [15]. For example, in [16] it is reported that reanalysis of data from the seven largest published microarray-based studies that have attempted to predict the prognosis of cancer patients reveals that five of those seven did not classify patients better than chance. To overcome this problem, until now the typical solutions were to select a correct validation method for estimating the classification error or to combine multiple datasets, but these two proposals have their problems and were not completely successful.

However, over the past few years, the price for sequencing DNA microarrays has been diminishing progressively, so it is to be expected that in the near future we can count on having significantly more samples. Increasing the number of samples would make DNA microarray analysis a less challenging domain (in the sense that it is now, regarding the high dimensionality characteristic), and would open the door for the use of techniques that were, until today, inapplicable. For example, the current paradigm when trying to classify DNA microarray data made almost paramount the

use of a dimensionality reduction technique as a preprocessing step, because having so many features (genes) for so few examples could produce overfitting of the models. But, paradoxically, some dimensionality reduction techniques such as wrappers could not have been applied either for the same reason. This issue will be further commented in the next subsection.

Another new line of research that can be opened in DNA microarray analysis—if it is possible to increase the sample size in the near future—is the use of more sophisticated methods such as those of deep learning. This discipline is revolutionizing, in the last few years, all the previous works in the field of machine learning, especially for images. Deep learning techniques have demonstrated to shown particular promising results in extracting high level abstractions from the raw data of very large, heterogeneous, high-dimensional datasets [17]. In the case of DNA microarrays, so far they were not large enough but, if this changes, deep learning methods would be a very interesting approach to deal with them (*see* Subheading 3.3) [18]. Besides, deep learning has led to a multimodality-based algorithm framework, enabling the effective fusion and use/retrieval of cross-domain multimedia data that could be used to obtain useful information from microarray data mixed with other information sources.

### 3.2 New Feature Selection Methods

Although microarray datasets contain usually very small samples for training and testing, the number of features in the raw data ranges in the order of thousands, since they measure the gene-expression en masse. To avoid the problem of the "curse of dimensionality" [19], feature (gene) selection plays a crucial role in DNA microarray analysis, which is defined as the process of identifying and removing irrelevant features from the training data (*see* Fig. 1), so that the learning algorithm focuses only on those aspects of the training data useful for analysis and future prediction [20]. There are usually three varieties of feature selection methods: filters, wrappers, and embedded methods. While wrapper models involve optimizing a predictor as part of the selection process, filter models rely on the general characteristics of the training data to select features independent of any predictor. The embedded methods generally use machine learning models for classification, and then an optimal subset of features is built by the classifier algorithm. Of course, the interaction with the classifier required by wrapper and embedded methods comes with an important computational burden (more important in the case of wrappers).

Feature selection as a preprocessing step to tackle microarray data has rapidly become indispensable among researchers, not only to remove redundant and irrelevant features, but also to help biologists identify the underlying mechanism that relates gene expression to diseases. Traditionally, the most employed gene selection methods fall into the filter approach, because of its low

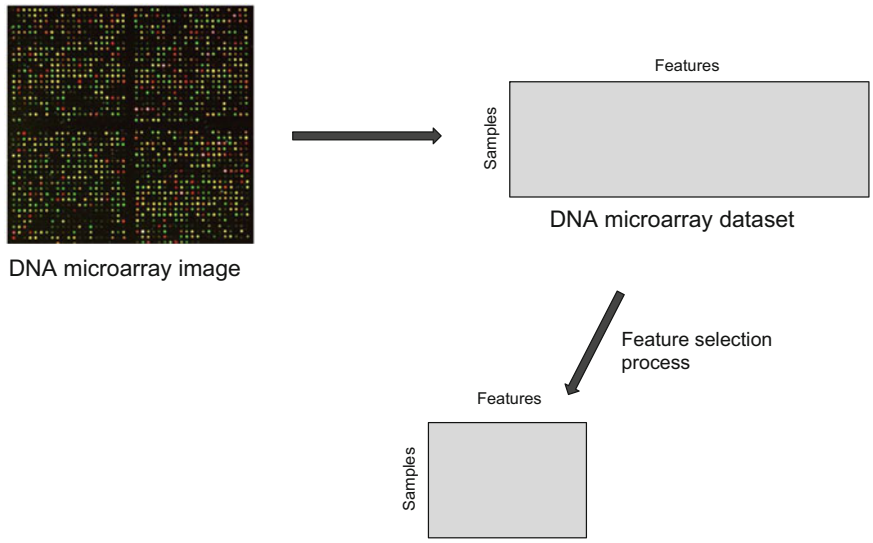


**Fig. 1** Scheme of a typical feature selection process on DNA microarray data

computational cost and the fact of having more features than samples. On the other hand, and due to the heavy computational consumption of resources and the high risk of overfitting, the wrapper approach has been largely avoided in the literature [21].

However, if in the near future and because of the cost decrease of DNA sequencing the number of samples is increasing, this could favor the use of more sophisticated feature selection methods. First of all, in this hypothetical scenario wrappers could be applied without incurring in overfitting of the data, with the benefit that they usually obtain better performance than filters.

Having large datasets in both number of samples and features might have the implication that most feature selection methods become impracticable, so they would need to be adapted. A possible solution is to use distributed feature selection methods, which take advantage of processing multiple subsets in sequence or concurrently [22]. This can be done in several ways. One of the possible approaches consists of distributing the data, running feature selection on each partition and then combining the results. The two main approaches to partitioned data distribution are by features (vertically) or by samples (horizontally). In [23] an exhaustive study of the implications of these two different distributions applied, among others, to microarray datasets is described. In a similar line, but aiming at deriving a more in-depth analysis of how distributed strategies affect specifically to DNA microarray data, the authors in [24] analyze several aspects of the distribution, such as the adequate number of nodes, the level of overlapping in vertical distribution, and the aggregation method used to join the

partial results./AQPlease check if edit made to the sentence, " One of the possible approaches consists..." is fine.

There are other platforms for performing distributed learning in which the distribution of the data is more transparent to the user. MapReduce [25] is one such popular programming model with an associated implementation for processing and generating large datasets with a parallel, distributed algorithm on a cluster. Hadoop, developed by Cutting and Cafarella in 2005 [26], is a set of algorithms for distributed storage and distributed processing of very large datasets on computer clusters; it is built from commodity hardware and has a processing part based on MapReduce. Developed more recently is Apache Spark [27], a fast, general engine for large-scale data processing, popular among machine learning researchers due to its suitability for iterative procedures. Although these paradigms are typically employed for parallelization of learning algorithms, they have not been so used when it comes to feature selection, offering an opportunity to parallelize new feature selection methods to be applied to microarray data. In this respect, some interesting contributions are those in [28–30], in which the authors adapt to the Spark platform some of the most well-known and employed feature selection methods.

Another option is the use of graphics processing units (GPUs) to distribute and thus accelerate calculations made in feature selection algorithms. With many applications to physics simulations, signal processing, financial modelling, neural networks, and countless other fields, parallel algorithms running on GPUs often achieve up to 100x speedup over similar CPU algorithms. It could be interesting to exploit GPU capabilities to design feature selection methods to cope effectively and accurately with thousands (or even millions) of genes. In [31] a package containing three different implementations of the mRMR—minimum redundancy maximum relevance—feature selection algorithm in several platforms, CPU for sequential execution (a faster version than the original one, introduced by Peng et al. in [32], and initially developed for dealing with the classification of DNA microarray data), GPU (graphics processing units) for parallel computing, and Apache Spark for distributed computing using big data technologies is presented.

Finally, a reasonable method to deal with the problem of large number of features and large-sample size is using subsampling. This statistical technique has been introduced by Politis and Romano [33] as an alternative procedure to classical resampling methods (such as the bootstrap) and it can speed up those with factors of thousands when the sample size is large. It consists in drawing resamples of a much smaller sample size from the original sample and using them to calibrate the sampling distribution of the statistic to be used. This is of course very useful for covariate significance test (e.g., to detect significant characteristics) or to compute confidence intervals for the probability of correct classification, among

many other. Interested readers are referred to the book by Politis et al. [34] for detailed explanation about the subsampling method. Subsampling has been used over the past few years for DNA data analysis. For instance, Aguirre de Cárcer et al. [35] have used subsampling methods for normalization of tagged high-throughput sequencing microbiomes datasets.

### 3.3 Deep Learning Techniques

Neural networks have been extensively used for years among machine learning researchers, in a "shallow" way (using a single of a few hidden layers). But, nowadays, the cutting-edge paradigm is to use *deep learning*, which consists basically in applying neural networks with many hierarchical layers of nonlinear information processing.

When it comes to process natural data in their raw form, standard machine learning techniques had the limitation of requiring considerable domain expertise to extract a feature vector that could be afterwards used by the learning method (typically a classifier). However, deep learning techniques remove this limitation, by being able to work directly with raw data and then discover the representations needed for detection or classification through composing simple but nonlinear modules that each transform the representation at one level into a representation at a higher, slightly more abstract level [36].

Although there already exist works that apply deep learning techniques to DNA gene expression [37, 38], this paradigm takes advantage of increases in the amount of available computation and data, so it is expected that better results are to be obtained if the sample size of DNA microarrays is increased. Besides, and as mentioned above, deep learning had made it possible to use multimodal algorithms that may take advantage of the fusion of different cross-domain multimedia data. In [18] the authors follow this line, and develop a sparse autoencoder method capable of learning a concise feature representation from unlabeled data. These unlabeled data can be obtained by combining data from different tumor cells provided that they are generated using the same microarray platform, and thus having more samples that can be used as a basis for feature learning.

Despite their broad benefits, deep learning models are often seen as black-boxes difficult to interpret, which implies a problem of user acceptance in critical sectors such as medicine, bioinformatics, or robotics. The next subsection will comment on the issue of interpretability.

### 3.4 Interpretability and Visualization

One of the main problems when using black-box techniques (such as deep learning methods or nonparametric methods) is interpretability of the results obtained. For instance, if our method is using hundreds or thousands of characteristics from our original dataset for a classification problem, biological interpretation of what the

method is doing is a difficult task. In that sense, those methods that select a limited number of features from a model are more useful for interpretation. One possible approach to limit the number of features of a method is to carry out some feature selection method based on covariate significance tests. In a fully nonparametric context Delgado and González-Manteiga [39] have proposed a covariate significance test for regression. This test is then a useful dimension reduction technique. However, when dealing with hundreds or thousands of characteristics, the test becomes unpractical without the use of statistical methods that account for false discovery rate. Combination of this method with already existing techniques to control the false discovery rate, such as those by Benjamini and Hochberg [40] and Benjamini and Yekutieli [41], provides a promising future in microarray data analysis with a large number of characteristics. Another line of research has also being opened recently on developing new visualization techniques for deep networks. The problem of visualizing, understanding, and interpreting deep neural networks has received a lot of attention lately, but the theoretical foundations of the interpretability problem are yet to be investigated and the usefulness of the proposed methods in practice still needs to be demonstrated, as it is the conclusion of the recent NIPS 2017 Workshop entitled "Interpreting, Explaining and Visualizing Deep Learning—Now what." There are several trends on this direction, as for example disentangling the chaotic representations of convulational layers into graphical or symbolic models. Another trend is the design of end-to-end learning interpreable neural networks, whose intermediate layers encode comprehensive patterns. Interesting surveys, more general or either centered on convolutional neural networks, explainability can be found in [42, 43], respectively.

## 4 Conclusions

Microarray data analysis has been traditionally a challenging, and as a consequence prolific domain for machine learning techniques. However, during the last years, new competitive high-throughput methodologies based on sequencing have appeared that are threatening to relegate microarray data analysis to the memory drawer, or at least severely limit its use. However factors that apply for other alternative technologies, such as economical costs, workflow complexities, or reproducibility in some contexts, among others still leave room for opportunities in microarray data analysis. Some new opportunities may arise from the hand of increasing sample sizes that in turn may derive new feature selection methods, or new implementations of well-known methods in parallel platforms like Spark, or GPU distributed approaches. Finally, the newly arrived deep learning techniques could add value to the field, as they allow

for the fusion of information coming from different sources. In this latter case, and although interpretability of the results can constitute a serious obstacle, during the very last years there have been new lines of research opened aimed at alleviating the opaqueness of the back-box representations of deep learning.

## References


1. Bumgarner R (2013) Unit 22.1 overview of DNA microarrays: types, applications, and their future. Curr Protoc Mol Biol 0:22
2. Zhao S, Fung-Leung WP, Bittner A, Ngo K, Liu X (2014) Comparison of RNA-seq and microarray in transcriptome profiling of activated T cells. PLoS One 9:e78644
3. Han Y, Gao S, Muegge K, Zhang W, Zhou B (2015) Advanced applications of RNA sequencing and challenges. Bioinf Biol Insights 9:29–46
4. McCarthy DJ, Campbell KR, Lun ATL, Wills QF (2017) Scater: pre-processing, quality control, normalization and visualization of single-cell RNA-seq data in R. Bioinformatics 33 (8):1179–1186
5. Everaert C, Luypaert M, Maag JLV, Cheng QX, Dinger ME, Hellemans J, Mestdagh P (2017) Benchmarking of RNA-sequencing analysis workflows using whole-transcriptome RT-qPCR expression data. Sci Rep 7:1559–1169
6. Conesa A, Madrigal P, Tarazona S, Gomez-Cabrero D, Cervera A, McPherson A, Szcześniak MW, Gaffney DJ, Elo LL, Zhang X, Mortazavi A (2016) A survey of best practices for RNA-seq data analysis. Genome Biol 17:13–31
7. Nazarov PV, Muller A, Kaoma T, Nicot N, Maximo C, Birembaut P, Tran NL, Dittmar G, Vallar L (2017) RNA sequencing and transcriptome arrays analyses show opposing results for alternative splicing in patient derived samples. BMC Genomics 18:443–460
8. Git A, Dvinge H, Salmon-Divon M, Osborne M, Kutter C, Hadfield J, Bertone P, Caldas C (2010) Systematic comparison of microarray profiling, real-time PCR, and next-generation sequencing technologies for measuring differential microRNA expression. RNA 16(5):991–1006
9. Hagemann IS (2016) Molecular testing in breast cancer: a guide to current practices. Arch Pathol Lab Med 140(4):815–824
10. Li X, Quigg RJ, Zhou J, Gu W, Rao PN, Reed EF (2008) Clinical utility of microarrays: Current status, existing challenges and future outlook. Curr Genomics 9:466–474
11. Chang KTE, Goytain A, Tucker T, Karsan A, Lee C-H, Nielsen TO, Ng TL (2018) Development and evaluation of a pan-sarcoma fusion gene detection assay using the NanoString nCounter platform. J Mol Diagn 20(1):63–77
12. Zhu H, Qian J (2012) Applications of functional protein microarrays in basic and clinical research. Adv Genet 79:123–155
13. Sauer U (2017) Analytical protein microarrays: Advancements towards clinical applications. Sensors 17(2):256–276
14. Dougherty ER (2001) Small sample issues for microarray-based classification. Comp Funct Genomics 2(1):28–34
15. Braga-Neto U (2007) Fads and fallacies in the name of small-sample microarray classification-a highlight of misunderstanding and erroneous usage in the applications of genomic signal processing. IEEE Signal Process Mag 24 (1):91–99
16. Michiels S, Koscielny S, Hill C (2005) Prediction of cancer outcome with microarrays: a multiple random validation strategy. Lancet 365(9458):488–492
17. Mamoshina P, Vieira A, Putin E, Zhavoronkov A (2016) Applications of deep learning in biomedicine. Mol Pharm 13(5):1445–1454
18. Fakoor R, Ladhack S, Nazi A, Huber M (2013) Using deep learning to enhance cancer diagnosis and classification. In: Proceedings of the 30th International conference on machine learning, ICML 2013. Journal of Machine Learning Research: W&CP, vol 28
19. Jain A, Zongker D (1997) Feature selection: evaluation, application, and small sample performance. IEEE Trans Pattern Anal Mach Intell 19(2):153–158
20. Guyon I, Gunn S, Nikravesh M, Zadeh LA (2006) Feature extraction: foundations and applications, vol. 207. Springer, Berlin
21. Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A, Benítez JM, Herrera F (2014) A review of microarray datasets and applied feature selection methods. Inf Sci 282:111–135
22. Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A (2015) Recent advances and emerging challenges of feature selection in the

context of big data. Knowl-Based Syst 86:33–45

23. Morán-Fernández L, Bolón-Canedo V, Alonso-Betanzos A (2017) Centralized vs. distributed feature selection methods based on data complexity measures. Knowl-Based Syst 117:27–45
24. Bolón-Canedo V, Sechidis K, Sánchez-Marono N, Alonso-Betanzos A, Brown G (2017) Exploring the consequences of distributed feature selection in DNA microarray data. In: Proc. International joint conference on neural networks, IJCNN2017, number IEEE Catalog number:CFP17-US-DVD. IEEE, Piscataway
25. Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. Commun ACM 51(1):107–113
26. Apache Hadoop (2018) http://hadoop.apache.org/ [Online; accessed Jan 2018]
27. Apache Spark (2018) https://spark.apache.org [Online; accessed Jan 2018]
28. Eiras-Franco C, Bolón-Canedo V, Ramos S, González-Domínguez J, Alonso-Betanzos A, Touriño J (2016) Multithreaded and spark parallelization of feature selection filters. J Comput Sci 17:609–619
29. Palma-Mendoza RJ, Rodriguez D, De-Marcos L (2018) Distributed Relief-based feature selection in Spark. Knowl Inf Syst 1–20
30. Ramírez-Gallego S, Mouriño-Talín H, Martínez-Rego D, Bolón-Canedo V, Benítez JM, Alonso-Betanzos A, Herrera F (2018) An information theory-based feature selection framework for big data under apache spark. IEEE Trans Syst Man Cybern Syst 48 (9):1441–1453
31. Ramírez-Gallego S, Lastra I, Martínez-Rego D, Bolón-Canedo V, Benítez JM, Herrera F, Alonso-Betanzos A (2017) Fast-mRMR: Fast minimum redundancy maximum relevance algorithm for high-dimensional big data. Int J Intell Syst 32:134–152
32. Peng H, Long F, Ding C (2005) Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans Pattern Anal Mach Intell 27:1226–1238
33. Politis DN, Romano JR (1994) Large sample confidence regions based on subsamples under minimal assumptions. Ann Stat 22:2031–2050
34. Politis DN, Romano JP, Wolf M (1999) Subsampling. Springer, Berlin
35. de Cárcer DA, Denman SE, McSweeney C, Morrison M (2011) Evaluation of subsampling-based normalization strategies for tagged high-throughput sequencing data sets from gut microbiomes. Appl Environ Microbiol 77(24):8795–8798
36. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444
37. Leung MKK, Xiong HY, Lee LJ, Frey BJ (2014) Deep learning of the tissue-regulated splicing code. Bioinformatics 30(12): i121–i129
38. Xiong HY, Alipanahi B, Lee LJ, Bretschneider H, Merico D, Yuen RKC, Hua Y, Gueroussov S, Najafabadi HS, Hughes TR *et al* (2015) The human splicing code reveals new insights into the genetic determinants of disease. Science 347(6218):1254806
39. Delgado MA, González-Manteiga W (2001) Significance testing in nonparametric regression based on the bootstrap. Ann Stat 29:1469–1507
40. Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. J R Stat Soc Ser B 57:289–300
41. Benjamini Y, Yekutieli D (2001) The control of the false discovery rate in multiple testing under dependency. Ann Stat 29:1165–1188
42. Chakraborty S, Tomsett R, Raghavendra R, Harborne D, Alzantot M, Cerutti F, Srivastava M, Preece AD, Julier S, Rao RM, Kelley TD, Braines D, Sensoy M, Willis CJ, Gurram P (2017) Interpretability of deep learning models: a survey of results. In: IEEE Smart World Congress 2017 Workshop: DAIS 2017 - workshop on distributed analytics infrastructure and algorithms for multi-organization federations. IEEE, Piscataway
43. Zhang Q, Zhu SC (2018) Visual interpretability for deep learning: a survey. eprint arXiv:1802.00614

# INDEX



Verónica Bolón-Canedo and Amparo Alonso-Betanzos (eds.), *Microarray Bioinformatics*, Methods in Molecular Biology, vol. 1986, https://doi.org/10.1007/978-1-4939-9442-7, © Springer Science+Business Media, LLC, part of Springer Nature 2019

## D



## E



## F



## G

## H



## I



## K



## L



## M

## N



## O



## P



## Q



## R

## S



## T



## U



## V