Statistical methods for data mining in genomics databases (Gene Set En- richment Analysis)

Konstantina Charmpi

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques Appliquées**

Arrêté ministériel : 7 août 2006

Présentée par

Konstantina Charmpi

Thèse dirigée par **Bernard Ycart**

préparée au sein **du Laboratoire Jean Kuntzmann**
et de **l'Ecole Doctorale ED-MSTII**

Méthodes statistiques pour la fouille de données dans les bases de données de génomique (Gene Set Enrichment Analysis)

Thèse soutenue publiquement le **3 Juillet 2015**,
devant le jury composé de :

M. Jean-Jacques Fournié
Directeur de recherche, CNRS, Toulouse, Examinateur
Mme Valentine Genon-Catalot
Professeure, Université Paris Descartes, Rapporteur
Mme Sophie Rousseaux
Directrice de recherche, INSERM, Grenoble, Examinatrice
Mme Adeline Samson
Professeure, Université Joseph Fourier, Présidente
M. Jacques van Helden
Professeur, Université Aix-Marseille, Rapporteur
M. Bernard Ycart
Professeur, Université Joseph Fourier, Directeur de thèse

Abstract

In different applications, such as cancer research, it is of particular interest to identify differentially expressed genes and/or groups of genes. This need has led to the development of many statistical methods, adapted to the treatment of genomic data. This work focuses on methods comparing a vector of numeric values, such as expression levels attached to each gene in the human genome, to a given set of genes, known to be associated for instance to some type of cancer, cellular function, or biological process. Acquisition methods for numeric data vectors are discussed, and a set of R functions realizing the main formatting operations is proposed. An overview of the existing statistical tests, from single-gene analysis to gene set analysis is given. Among those, Gene Set Enrichment Analysis (GSEA) is probably the most basic tool for genomic data treatment. However, from a statistical point of view, the centering of its test statistic does not allow the derivation of asymptotic results. Moreover, the calculation of p-values is a very time-consuming, low precision procedure. A test statistic with a different centering is proposed. Under the null hypothesis, the convergence in distribution of the new test statistic is proved, using the theory of empirical processes. The limiting distribution needs to be computed only once, and can then be used for computing the p-values of many different gene sets. This results in very large savings in computing time. The test defined in this way has been called Weighted Kolmogorov Smirnov (WKS) test, since it can be viewed as a generalization of the classical Kolmogorov-Smirnov goodness-of-fit test.

Next, a different problem is addressed: the large number of false discoveries from existing methods. Based on a statistical study of several databases of gene sets, coupled with another study of a large number of expression vectors, an explanation is proposed: the null hypothesis of existing methods, which is that all genes have the same probability to be included in a gene set, is far from true in practice. A generalization of the WKS test, called Doubly Weighted Kolmogorov Smirnov (DWKS) test is proposed: it is based on a differential weighting of the genes, taking into account their relative frequencies in real data. The two tests (WKS and DWKS) have been applied both to simulated and real data, and compared with other existing procedures. Using expression data from the GEO repository, tested against the MSig Databases, a comparison between the GSEA test and the new procedures has been conducted. Our conclusion is that, beyond their mathematical and algorithmic advantages, the WKS and DWKS tests could be more informative in many cases, than the classical GSEA test and efficiently address the issues that have led to their construction.

Résumé

Dans différents domaines d'application, tels que la cancérologie, il est crucial d'identifier des gènes ou des groupes de gènes, significativement sur- ou sous-exprimés. Ce besoin a conduit au développement de nombreuses méthodes statistiques, adaptées au traitement des données de génomique. Ce travail est centré sur les méthodes visant à comparer un vecteur de données numériques, telles que des niveaux d'expression liés à chacun des gènes du génome humain, à un ensemble donné de gènes, connus pour être associés par exemple à un type de cancer, à une fonction cellulaire, ou à un processus biologique. Les méthodes d'acquisition des vecteurs de données sont discutées, et un ensemble de fonctions R, réalisant les opérations de formatage principales, a été implémenté. Une revue des tests statistiques traitant les gènes individuellement ou par groupes, est proposée. Parmi ces méthodes, le test Gene Set Enrichment Analysis (GSEA) est probablement le plus largement utilisé pour le traitement des données de génomique. Néanmoins, du point de vue statistique, son centrage ne permet pas l'établissement de résultats asymptotiques. De plus, le calcul des p-valeurs est algorithmiquement coûteux, et peu précis. Une statistique de test centrée différemment est proposée. Sous l'hypothèse nulle, la convergence en loi de la nouvelle statistique de test est démontrée, en utilisant la théorie des processus empiriques. La loi limite est à calculer une seule fois, et peut ensuite être utilisée pour calculer la p-valeur d'ensembles de gènes différents. Ceci se traduit par une économie importante en temps de calcul. Le test ainsi défini est appelé test de Kolmogorov-Smirnov pondéré, car on peut le voir come une généralisation du test d'ajustement de Kolmogorov-Smirnov classique.

Un autre problème est abordé: le grand nombre de fausses détections par les méthodes existantes. À partir d'une étude statistique de plusieurs bases de données d'ensembles de gènes, couplée à une autre étude sur un grand nombre de vecteurs d'expression, une explication est proposée: l'hypothèse nulle des méthodes existantes, qui stipule que tous les gènes ont la même probabilité d'être inclus dans un ensemble de gènes, est loin d'être vérifiée en pratique. Une généralisation du test précédent, baptisée test de Kolmogorov-Smirnov doublement pondéré, est proposée. Elle est basée sur une pondération des gènes, qui prend en compte leurs fréquences relatives dans les données réelles. Les deux tests ont été appliqués à des données simulées et réelles, et leurs résultats comparés aux procédures existantes. À partir de données cliniques de la base GEO, testées contre les ensembles de gènes MSig, une comparaison entre le test GSEA et les nouvelles procédures a été menée. Notre conclusion est que, au-delà leurs avantages mathématiques et algorithmiques, les tests proposés pourraient se révéler, dans de nombreux cas, plus informatifs que le test GSEA classique, et traiter efficacement les deux problèmes qui ont motivé leur construction.

Acknowledgements

First of all, I would like to thank my supervisor, Mr Bernard Ycart for providing me with his full support and continuing guidance throughout the whole period I have been preparing my thesis. At this point, I would like to describe shortly the events that led me here in Grenoble: after three "difficult" months of sending unsuccessfully applications in universities in the whole France and when I started to lose every hope to do my PhD there, a mail on Saturday, June 30 2013, came to change my life. Thus, before all, I feel an immense gratitude to him for being the only person to have given me this chance. Next, I would like to emphasize on his great kindness and generosity by helping me in every aspect of my everyday life here. Regarding the thesis itself, I feel the need to thank him for having spent so much time with me, not only with our very regular (usually weekly) meetings but also with accepting to see me every time I needed so. Equally important is the fact that he did not accept me being overtimid and insisted on the fact that apart from the writing skills, I should also communicate and present my ideas orally. He is for sure the best mathematician I have ever met and I admire so much his very fast way of thinking. I would like to thank him more than anything for having supervised my thesis and for all the knowledge he provided me with.

Next, I would like to mention that spending one month and a half in the Cancer Research Center of Toulouse (CRCT) was a unique experience for me. There, I had the chance to collaborate with great biologists and clinicians, strenghten my background in biology and work with real data. However, I would like to distinguish the director of CRCT, Mr Jean-Jacques Fournié. He is the best director I have ever met and I could never think of someone more suitable for that position. He never gets tired, he is enthusiastic, always willing to meet and discuss with everyone and share his knowledge. He gives many chances and motivation to young people as well. I would like to thank him so much for his great support and belief in me.

Then, I would like to address a big thanks to Mrs Sophie Rousseaux for organizing the seminars of the club bioinfo, where I had the chance to attend and learn many new things lying more in biology and bioinformatics from experts of the domain. I would also like to thank her equally for the very important times where she shared her knowledge with me.

In addition, I would like to thank very much the two reviewers, Mrs Valentine Genon-Catalot and Mr Jacques van Helden who took the time to read my manuscript, evaluate it and make very useful suggestions and corrections. I would also like to thank Mrs Adeline Samson who accepted to be part of the jury.

I would also like to thank my colleagues and in particular my friends Nhu and Patricia for sharing the experience of being a PhD student and being next to me whenever I needed them. An equally big thanks goes to Cecilia, that despite her limited stay in Grenoble, her company was of an equal importance to me. What is more, I would like to thank Alexandre for answering all my questions.

Moreover, I would like to thank very warmly the secretary of LJK, Mrs Hélène Baum, whose help has been continuous and invaluable to me and started before I came here in Grenoble.

Last but most importantly, I would like to thank my family, to which I owe everything. I would like to thank them for their unconditional help and support in every step of mine and their moral encouragement in the achievement of my goals. One is for sure: without them, I wouldn't have been here.

Contents

List of Abbreviations

DWKS	Doubly Weighted Kolmogorov Smirnov
FCS	Functional Class Scoring
FDR	False discovery rate
FE	Fisher's exact
FWER	Family-wise error rate
GAGE	Generally Applicable Gene Set Enrichment
GEO	Gene Expression Omnibus
GSEA	Gene Set Enrichment Analysis
IM	Ideal Mismatch
IQR	Interquartile range
KS	Kolmogorov Smirnov
MAD	Median absolute deviation
MAS	Microarray Suite
MBEI	Model Based Expression Index
MEA	Modular Enrichment Analysis
MGF	Median gene frequency
MM	Mismatch
MS	Median gene set size
MSig	Molecular Signature
NbGS	Number of gene sets
NCBI	National Center for Bioinformatics
NG	Number of different genes
NGS	Next Generation Sequencing
NSB	Nonspecific binding
PAGE	Parametric Analysis of Gene Set Enrichment
PM	Perfect Match

RMA	Robust Multichip Average
SAGE	Serial Analysis of Gene Expression
SEA	Singular Enrichment Analysis
WKS	Weighted Kolmogorov Smirnov
ZE	Zelen's exact

Chapter 1

Introduction

Cancer is considered as the number one cause of death worldwide. The most common types in males are lung, prostate, colorectal, and stomach cancers; and in females breast, colorectal, lung, and cervical cancers. In children, acute lymphoblastic leukaemia and brain tumors are the most frequent types. Cancer can be regarded as a genetic disease occurring as a result of progressive accumulation of genetic misregulations that lead to the aberrant behavior common to all cancer cells: dysregulated growth, lack of contact inhibition, genomic instability, and propensity for metastasis. Fortunately, the continuous development of technology and the extensive research carried out in the domain have led to significant progress: medicines that cure or inhibit the progress of cancer have been produced, life expectation of patients has been notably improved, long-term remissions are routinely obtained. Yet, with few exceptions, cancer remains an incurable disease, and a lot of research must still be done, in particular concerning the epigenetic aspects of the disease [33].

Fortunately, powerful techniques for collecting and interpreting genetic data are now available. Many of those techniques aim at evaluating the quantity of messenger RNA produced by a given gene in a time unit, which is referred to as its expression level [38]. The study of the transcriptome requires the measurement of the expression levels of thousands of genes simultaneously. This is now achieved by high-throughput methods. Microarray is a high-throughput technology that measures the expression levels of thousands of genes at once [98, 79]. Microarrays have become the standard tool for understanding gene functions, regulations and interactions. This technique is not limited to cancer research. Microarrays have a major impact on biomedical research that increases our understanding of all aspects of human diseases. Even though microarray data will be our main focus here, other existing techniques eventually associate numeric data to the set of (nearly) all genes in the human genome: Serial Analysis of Gene Expression (SAGE) [14] and next generation sequencing (NGS) that becomes more and more dominant [129]. All these techniques end up producing, for one given sample of tissue, what will be considered here as a vector of numbers, the entries of which are indexed by gene symbols. Such *vectors* are our main object of study. Their entries may be interpreted as expression levels, but might also consist of p-values, correlations, fold-changes, t-statistics, signal-to-noise ratios, etc. In order to keep a sufficient degree of generality, the entry of a vector associated to any given gene will be referred to as its *weight*. Many examples of such data can be downloaded from the Gene Expression Omnibus (GEO) repository [43].

The procedure from the implementation of the biological experiment, to the acquisition of a vector of weights is far from straightforward. Extracting the useful information on gene expression from the available output is not trivial. The abundance of data offers many possibilities, but at the same time opens big challenges: rigorous procedures for analysis are required. First of all, the data collection process is quite noisy, and non-biological bias may be introduced at a number of points by the operators or by the technology. Thus, normalization methods have to take place in order to remove the noise from the microarray data. Then the standard treatments of annotation (mapping from probes to gene symbols) and reduction (removal of duplicate gene symbols) need to be performed. We have developed our own set of R functions to download GEO datasets, perform various pre-treatments on those datasets, and output matrices of vectors together with biological information about those

vectors [125].

Once vectors of weights have been calculated, one major goal may consist for instance in identifying differentially expressed genes, i.e. genes whose expressions differ between two or more conditions (e.g. healthy vs. cancerous cells, different treatments, good and bad survival, etc.). At first, available measures and statistical tests treated each gene separately (single-gene analysis). Then, the rapidly increasing wealth of gene set databases in Bioinformatics offered another direction of research. Groups of genes associated to a given biological process, a particular location in the cell, a molecular function, a type of cancer, etc. were defined, thus summarizing biological knowledge from advances in the literature. Such *gene sets* are the other main data type considered here. In the applications that will be proposed, the gene sets come from thematic lists from the Molecular Signature (MSig) database [108]. Our procedures are applicable to other gene sets, including those deduced from vectors by thresholding: for instance the set of all genes corresponding to the highest or lowest percentiles of a vector.

Given a vector and a gene set, the question to be answered can be stated as follows: is the distribution of weights of those genes which are inside the gene set significantly different from the distribution of weights outside the gene set? Implicitly or not, existing methods consider as a null hypothesis that the gene set is a random subset of the set of all genes, each gene being included in the subset with equal probability. As customary in statistical testing, three alternatives to the null hypothesis can be considered, corresponding here to the gene set having significantly higher weights (enrichment), lower weights (depletion) or both (two-tailed test). A large number of statistical tests comparing vectors and gene sets has been developed [68, 108, 70, 36, 24, 81, 64]. These tests can be divided into three classes: singular enrichment analysis (SEA), functional class scoring (FCS) and modular enrichment analysis (MEA). Undoubtedly, among this great variety of statistical tests, Gene Set Enrichment Analysis (GSEA) [108], belonging to the second class, is one of the most successful tools used for genomic data treatment. However, it is also a common fact that this procedure is very time-consuming, and insufficiently precise, since the distribution of the test statistic has to be estimated through Monte-Carlo simulation for every gene set. In addition, we have observed that the test statistic in GSEA is not the most accurate from a theoretical perspective, since its centering is not really appropriate; to be more exact it does not allow the derivation of asymptotic results. For this reason, a test statistic with a different centering has been proposed and studied. The test defined in this way has been called Weighted Kolmogorov Smirnov (WKS) test [26], since it can be interpreted as a generalization of the classical Kolmogorov-Smirnov goodness-of-fit test. This is the main theoretical input of this work.

A different problem has also been studied. Existing statistical procedures output a large number of false discoveries, in the sense that when testing virtually any vector of actual data against a given database of gene sets, several thousand gene sets are usually declared significant, including a majority that have no biological relevance. We argue that this mainly comes from the fact that the null hypothesis of current tests is not appropriate: genes have quite different probabilities to be included in an actual gene set, in contradiction with the equal probability null hypothesis. Indeed for a typical list such as those of the MSig database, the vast majority of genes are only present in a few gene sets whereas several hundred genes are present in dozens of gene sets. Moreover, after analyzing a large numbers of vectors, we have observed that the most frequent genes typically had higher weights than the unfrequent ones. This explains why a large number of gene sets may appear as enriched against a given vector, independently from their biological interest. We propose a generalization of the WKS test that includes non uniform null hypotheses on the gene sets. It has been called Doubly Weighted Kolmogorov Smirnov (DWKS) test, and constitutes the second theoretical input of the thesis. The DWKS test addresses the problem of the excessive false discovery rate, by taking into account the different frequencies of the genes in the database. The validation of our testing procedures was carried out both theoretically on simulated data, and practically on GEO datasets tested against the MSig database. The procedures have been implemented in the statistical language R, and included in the more user-friendly software AutoCompare.

The thesis is organized in the following way. Chapter 2 proposes a general review. Firstly, some needed notions of Bioinformatics are recalled, beginning with the basic terminology of genetics in

subsection 2.1.1, continuing with the description of the microarray technology in subsection 2.1.2, and ending with the presentation of several widely used gene set databases in subsection 2.1.3. The normalization methods used to remove the noise from microarray data are also reported in that section. In the next section, the treatments of annotation and reduction that must be applied to raw microarray data in order to format them for further statistical analysis, are described in subsection 2.2.1. Then the problem of formatting and merging datasets coming from different studies is addressed in subsection 2.2.2. In the last part of Chapter 2, an overview of existing statistical tests aiming at identifying differentially expressed genes or groups of genes, is presented: single-gene methods in subsection 2.3.1, multiple testing in subsection 2.3.2, then the main SEA methods in subsection 2.3.3. A special emphasis is put on GSEA, a FCS method: after presenting the method in subsection 2.3.4, its drawbacks are detailed in subsection 2.3.5. Other FCS methods are presented in subsection 2.3.6, then MEA methods in subsection 2.3.7.

Chapter 3 contains our theoretical results about the WKS and DWKS tests. The main theoretical result is Theorem 3.1.1, which proves the convergence of some empirical processes to a stochastic integral process. The convergence in distribution for the two-sided test statistic is deduced. The generalization to the DWKS test is treated in subsection 3.1.3. The consistency of both tests is proved in subsection 3.1.4. In the rest of section 3.1, some theoretical properties of the limiting process are studied: it is not Markovian (subsection 3.1.5), it is time reversible on $(0, 1)$ (subsection 3.1.6), explicit calculations for a step weight function are feasible (subsection 3.1.7). Finally, tail estimates for the test statistic distribution under the null hypothesis are presented in subsection 3.1.8. Section 3.2 discusses the actual implementation of both tests, at first in their two-sided versions (subsections 3.2.1 and 3.2.2), then for one-sided versions (subsection 3.2.3). A discussion of the choices to be made by the user in real applications comes in subsection 3.2.4. In section 3.3, theoretical validation on simulated data is presented. Sample sizes for which the asymptotic approximation is legitimate, are studied in subsection 3.3.1. Finally, explicit theoretical results in two particular cases are validated on simulated data in subsection 3.3.2.

Chapter 4 is devoted to applications. In the first two sections, statistical results on large numbers of gene sets (section 4.1) and vectors (section 4.2) are presented. They support our analysis of the causes for the excessive false discovery rate of existing methods. In section 4.3, existing and proposed statistical tests are compared in terms of power, over simulated data under different alternatives. Then in section 4.4, it is shown that the DWKS test actually solves the false discovery problem. A comparison of the WKS and DWKS tests to the classical GSEA method over real data from the GEO repository, is made in section 4.5. Our main conclusion is that the new tests actually decrease false discoveries, while biological meaning is preserved. Chapter 4 ends with a presentation of the AutoCompare software that has been developed from our theoretical results. The interface and functionalities are presented in subsection 4.6.1. A manual on how to use the R codes for data formatting (subsection 4.6.2) and (D)WKS testing (subsection 4.6.3) is included. The R codes themselves are given in the Appendix.

Finally, in Chapter 5, a general conclusion is drawn and several perspectives of the current work are discussed.

Chapter 2

Overview of existing methods

2.1 Biological review

The two types of statistical variables that are considered in this work are vectors of numeric values indexed by a set of genes, and gene sets, or pathways. In this section, the construction of such data is described.

2.1.1 Basic terms in biology

DNA is a sequence of nucleotides in a double helix. Each nucleotide consists of a base, a sugar and a phosphate. In DNA, there are four bases: adenine (A), cytosine (C), guanine (G) and thymine (T). Base-pairing occurs according to the following rule: G pairs with C and A pairs with T. A *gene* is a part of the DNA carrying important information for the organism. *Gene expression* is the process by which information from a gene is used in the synthesis of a functional gene product. These products are often proteins. The gene expression usually comprises two steps: *transcription*, where the conversion of the information from the gene into messenger RNA (mRNA) takes place and *translation*, where mRNA is converted into protein, which is responsible for a cell function. mRNA is single-stranded and has uracil (U) instead of T as one of the bases. The complete set of all genes being transcribed is referred to as the *transcriptome*.

The more active a gene is, the more mRNA it produces. It is actually the level of gene expression (number of mRNA transcripts) that distinguishes physiological status (nutrition, environment), sex and age, various tissues and cell types, response to stimuli (drugs, signals, toxins), health and disease, including underlying pathogenic diversity, progression and response to treatment and patient classes of varying prospects.

Thus, the need for measuring gene expression becomes imperative. This is done by use of microarrays that will be analyzed in the next subsection. Once having the measurements, questions such as the following can be addressed:

- What are the differences (in gene expression) between cell lines?
- What is the difference between a tumor and a healthy tissue?
- Are there different tumor types?
- What genes are the key elements in a biological process?

More information on biological issues can be found in numerous textbooks about molecular biology (see for instance [29] or [101]).

2.1.2 Measuring gene expression

The early gene expression measurement methods, such as the northern blotting and the quantative PCR were low-throughput methods. Typically, measures were done for one gene (or a small number of genes) at a time. Since this was not so practical, these methods were soon replaced by high-throughput methods. Microarray is a high-throughput technology that measures the expression levels of thousands of genes at once. There are two major types of microarrays:

- single-color Affymetrix genechips [79] and
- two-color cDNA or oligo arrays [98]

The underlying principle of microarray technology is the ability of DNA to bind to itself and to RNA. Affymetrix and cDNA arrays are conceptually similar, with differences in manufacture and details of design and analysis.

The first type consists of silicon chips, with oligonucleotide probes lithographically synthesized on the array. An example of Affymetrix genechip is shown on Figure 2.1. The chip has size $1,28 \times 1,28$ centimeters. Each genechip array has typically hundreds of thousands of locations, called probe cells (or probes or spots). Every probe cell has millions of DNA strands built up and every strand usually consists of 25 bases. A probe set is a group of probes that correspond to a particular gene or expressed sequence tag (EST). Most genes or ESTs are represented by a single probe set, but some are represented by more than one. The probe set consists of probe pairs (usually $11 - 20$). A probe pair consists of a:

- Perfect match (PM) probe which exactly represents part of the DNA sequence for a gene or EST
- Mismatch (MM) probe which differs from the PM probe only at the middle base (A↔T, C↔G).

On the array, the MM probe is located directly below its corresponding PM probe.

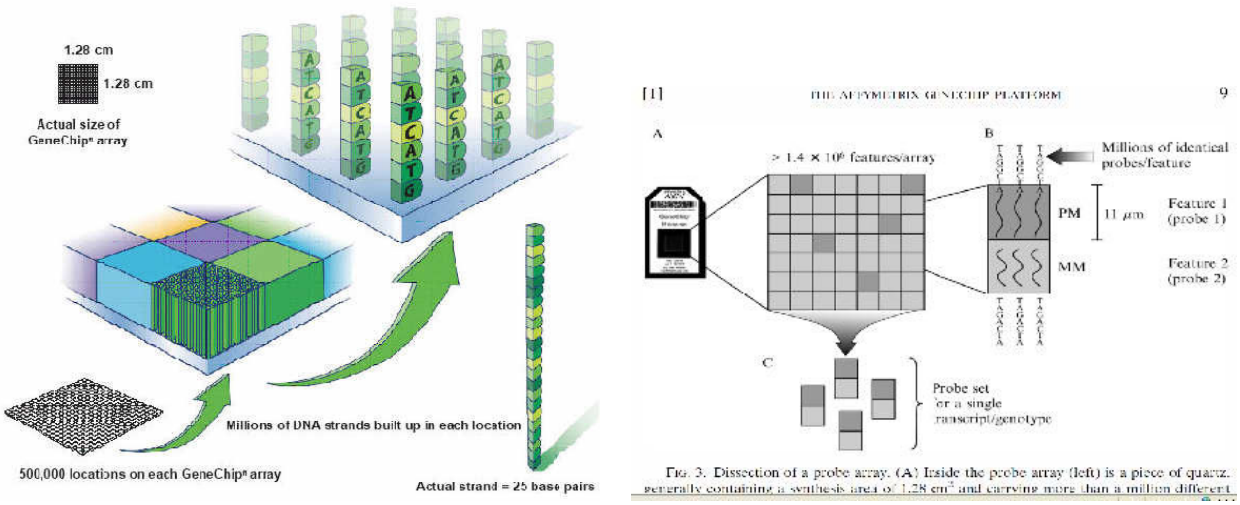


Figure 2.1: Affymetrix GeneChip array example: detailed schematic representation of a probe cell (left) (available at `http://www.vsni.co.uk/software/genstat/htmlhelp/marray/AffymetrixChips.htm`) and of a probe pair (right) [25].

The Affymetrix GeneChip arrays are one-color arrays and thus each array measures mRNA-abundance for one tissue sample only. When preparing the biopsies or tissue samples for which the mRNA levels are to be analyzed, a number of laboratory steps need to be performed. Firstly, mRNA samples are obtained under all experimental conditions. Then, single stranded cDNA is synthesized by reverse transcription. Next, the single stranded cDNA is converted into double stranded cDNA.

An in vitro transcription is then carried out to produce biotin-labeled cRNA. The resulting cRNA is fragmented and hybridized to the Affymetrix Gene Chip array. After removing the non-hybridized cRNA, a scanner is used to create an image of the array. The raw image data are saved in a DAT file. The information about each probe on the chip is extracted from the image data by the Affymetrix image analysis software. The information is stored in the CEL file. The intensity of each "spot" indicates how much binding has occurred at that spot. Intensity readings from PM probes represent gene specific binding (GSB) plus some binding due to cross hybridization (nonspecific binding). Intensity readings from MM probes can be used to account for nonspecific binding (NSB). More detailed information on all steps required before attaining the image file can be found in the Expression Analysis Technical Manual [4].

The actual preprocessing of microarray starts with the CEL files and usually consists of three steps aiming at removing non biological elements of the signal:

- *Background correction* (adjustment)
- *Normalization*
- *Summarization* (aggregation)

A *background correction* is usually performed to adjust for optical noise, that is intensity not related to hybridization. Sometimes, a NSB correction is performed as part of the procedure. *Normalization* is used to correct for systematic array differences. Finally, *summarization* consists in combining levels of corresponding PM and MM probes to a single expression measure for each probe set.

In BioConductor [50], which is an open-source and open-development software project built in the R statistical programming environment [94] for the analysis and comprehension of genomic data, many methods are available, combining all three steps above and are used as a first step in microarray data analysis. Some of the most widely used methods are Affymetrix's Microarray Suite (MAS) 5 [3], Model Based Expression Index (MBEI) implemented through dChip (`www.dchip.org`) [75], Robust Multichip Average (RMA) [63] and RMA using sequence information (GCRMA) [123].

The aforementioned methods will now be described briefly. More details can be found in the references. For a fixed probe set, let PM_{ij} and MM_{ij} denote the PM and MM intensity values of the i-th array and j-th probe pair in a total of I arrays and J probe pairs.

MAS5

1. Background correction: $E_{ij} = \log_2(PM_{ij} - IM_{ij})$ for a fixed probe set, array i and probe j.
2. Normalization: Scale so that the trimmed mean of the E_{ij} values is the same for each chip.
3. Summarization
$$\log_2(S_i) = TukeyBiweight(N(E_{i1}), \ldots, N(E_{iJ}))$$
The Tukey biweight algorithm ([3],[30, p. 124-127]) is a method to determine a robust average unaffected by outliers.

   Probe Set Summary = S_i,

   where IM (Ideal Mismatch) is a quantity derived from the MMs that is never bigger than its PM pair ($IM < PM$) (see `http://media.affymetrix.com/support/technical/whitepapers/sadd_whitepaper.pdf`) and $N(\cdot)$ denotes the normalized value.

dChip/MBEI

1. Background correction: $E_{ij} = PM_{ij} - MM_{ij}$ for a fixed probe set, array i and probe j.
2. Normalization: A normalization curve is fit to a group of PM probes which are thought to be unchanged (invariant set normalization).

3. Summarization: Fit $N(E_{ij}) = \mu_i a_j + \epsilon_{ij}$

   Probe Set Summary = estimated μ_i,

   where μ_i's and a_j's represent the array and probe effects, $N(\cdot)$ denotes the normalized value and ϵ_{ij}'s are random normal errors with zero mean.

RMA

1. Background correction: $E_{ij} = PM_{ij} - B_{ij}$ for a fixed probe set, array i and probe j, where B is estimated so $B < PM$ (see [63]).

2. Normalization: Forces distribution of PM values to be the same for every array in an experiment (Quantile normalization). More precisely, the normalization maps probe level data from all arrays, $i = 1, \ldots, I$, so that an I-dimensional quantile-quantile plot follows the I-dimensional identity line: see Bolstad et al. [15] for more details.

3. Summarization: Robustly fit a two-way anova with array- and probe-effects $\log_2(N(E_{ij})) = \mu_i + a_j + \epsilon_{ij}$

   Probe Set Summary = estimated μ_i,

   where μ_i's and a_j's represent the array and probe effects, $N(\cdot)$ denotes the normalized value and ϵ_{ij}'s are i.i.d. with zero mean.

GCRMA

1. Background correction: Wu et al. [123] note that "RMA does not adjust well for nonspecific binding" and thus incorporate probe sequence to better estimate NSB. More specifically, probe affinity is modeled as a sum of position-dependent base effects, and can thus be calculated for each PM and MM value, based on its corresponding sequence information. More information can be found in [123], [106, p. 48-49] and `https://www.chem.agilent.com/cag/bsp/products/gsgx/downloads/pdf/GCRMA_Probe_Summarization.pdf`.

2. Normalization: same as RMA.

3. Summarization: same as RMA.

It should be noted that RMA and GCRMA return $\log_2$ transformed data, while MAS5 and MBEI return raw data. Furthermore, it is obvious from the previous description that MAS5 can only be used on a single array, while RMA, GCRMA and MBEI can be used on a set of arrays. Most of the above normalization methods are implemented in the R/Bioconductor package affy [49].

It is worth mentioning that in the preprocessing procedure, it is necessary to use the Chip Description File (CDF), which, among other things, describes probe locations and probe set groupings on the chip.

A question that naturally arises is: which method should be chosen? In fact, there is no best normalization method. The selection of an appropriate method relies on the intention of the study. It is important to balance between accuracy and precision (bias variance trade off). However, the R/Bioconductor package affycomp [65] helps comparing the different normalization procedures. More specifically, the affycomp package provides many tools for assessing preprocessing algorithms. These tools benchmark different characteristics of the resulting expression measures. The assessments can be visualized as plots or summary statistics.

The second type of microarray consists of glass slides or similar supports containing cDNA sequences that serve as probes for measuring mRNA levels in samples. cDNAs are arrayed on each slide in a grid (aka. print-tip group, sector) of spots (probes). Each spot contains thousands of copies of a sequence that matches a segment of a gene's coding sequence, whose length may vary from a few hundred bases to a thousand. Different spots typically represent different genes but some genes may be represented by multiple spots. The spotted sequences are known (or can be determined). Their locations on the array are also known and do not change from slide to slide. A single slide typically

contains thousands of spots. mRNAs are extracted from samples of interest, reverse transcribed into cDNAs and labeled with a fluorescent dye. Usually two samples, dyed with different dyes, are hybridized to a single slide. Cyanine 3 (Cy3, green dye) and Cyanine 5 (Cy5, red dye) are currently the two most commonly used dyes. A laser excites the dye and a scanner records an image of the slide. The raw data from a cDNA microarray experiment consist of pairs of image files, 16-bit TIFFs, one for each of the dyes. It is common to superimpose the two images. Then a yellow spot indicates similar expression levels in both samples, a green spot indicates overexpression of the gene in the sample labeled with Cy3, a red spot indicates overexpression of the gene in the sample labeled with Cy5, while a black spot means no expression of the gene in any of the two samples. A result of overlay of the two images is depicted on Figure 2.2. The TIFF images are then processed using an image analysis program, such as ArrayVision, ImaGene, GenePix, QuantArray or SPOT to acquire the red and green foreground and background intensities for each spot. The spot intensities are then exported from the image analysis program into a series of text files. There should be one file for each array, or in the case of ImaGene, two files for each array.

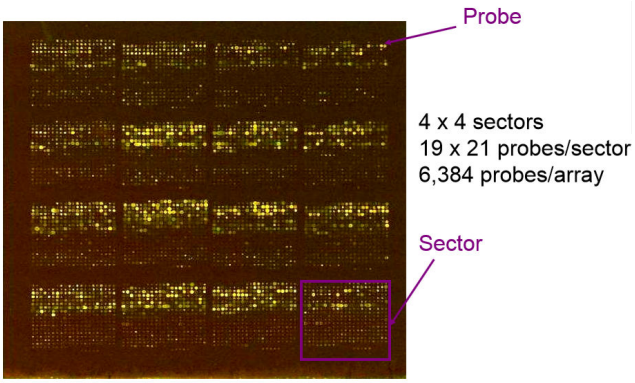


Figure 2.2: Overlay of two images [39]

The normalization procedure for two-color arrays is quite different. Here, we can distinguish three types of normalization:

- Within-slide normalization
- Paired-slides (dye swap)
- Between-slides normalization

Let R and G denote the background-corrected red and green intensities for each spot. Background correction consists usually in subtracting the background from the foreground intensities. Instead of considering R vs G or $\log_2 R$ vs $\log_2 G$, it is preferable to use $M = \log_2 R - \log_2 G$ vs $A = (\log_2 R + \log_2 G)/2$. An MA-plot amounts to a 45^o counterclockwise rotation of a $\log_2 R$ vs $\log_2 G$ plot followed by scaling.

The within-slide normalization aims at balancing red and green intensities. The imbalances can be caused by different incorporation of dyes, different amounts of mRNA and different scanning parameters. In practice, there is usually the need to increase a bit the red intensity so that it balances the green. Some popular methods for within-slide normalization are listed below:

1. Global normalization

   Methodology

$$\log_2\left(\frac{R}{G}\right) \to \log_2\left(\frac{R}{G}\right) - c = \log_2\left(\frac{R}{kG}\right) ,$$

where the location parameter $c = \log_2(k)$, usually the mean of the log-ratios M, is constant across the spots. Instead of the mean, the median can be used in cases of aberrant gene intensities, since it is more robust to outliers, or the trimmed mean when high and low extreme intensities are present, since the top and bottom $n\%$ values are excluded from the calculation of the array mean.

2. Intensity-dependent normalization: Here, the assumption that dye bias is dependent upon spot intensity is made.

   Methodology
   $$\log_2\left(\frac{R}{G}\right) \to \log_2\left(\frac{R}{G} - c(A)\right) = \log_2\left(\frac{R}{k(A)G}\right) .$$
   One estimate of $c(A)$ is made using the LOWESS function of Cleveland [28]: LOcally WEighted Scatterplot Smoothing.

3. Intensity and sector-dependent (within-print-tip-group) normalization: In addition to intensity-dependent variation in log ratios, spatial bias can also be a significant source of systematic error. Most normalization methods do not correct for spatial effects produced by hybridization artefacts or print-tip or plate effects during the construction of the microarrays. It is possible to correct for both print-tip and intensity-dependent bias by performing LOWESS fits to the data within print-tip groups, i.e.
   $$\log_2\left(\frac{R}{G}\right) \to \log_2\left(\frac{R}{G} - c_i(A)\right) = \log_2\left(\frac{R}{k_i(A)G}\right) ,$$
   where $c_i(A)$ is the LOWESS fit to the MA-plot for the i-th grid only.

Figure 2.3 depicts an MA-plot (typical of what is obtained in cDNA microarray experiments) together with the within-print-tip-group lowess fits.

Sometimes, in the last case (within-print-tip-group normalization), some scale adjustment may also be necessary. For each sector, scale by the median absolute deviation (MAD) (for more details see [103]). However, in general, further normalization like this one, should be applied only when diagnostic plots show strong evidence of the need for such normalization, since unnecessary estimation and removal of trends adds noise to the data.

After normalization of the M-values for each array, the red and green distributions become essentially the same for each array, but there might be some variation between arrays. In such a case, applying quantile normalization to the A-values makes the distributions essentially the same across arrays as well. Applying quantile normalization directly to the individual red and green intensities produces a similar result. Moreover, a scale-normalization (extension of within-slide scale normalization), consisting in a simple scaling of M-values from a series of arrays, so that each array has the same MAD, could be equally employed.

All the above techniques are offered in the R/Bioconductor package limma [104]. An overview of the normalization techniques for two-colour arrays is given by Smyth and Speed in [103], where they also describe how the methods are implemented in the limma package, including choices of tuning parameters.

Finally, when many genes are expected to change, then a paired-slides normalization should be tried. Perform two hybridizations with dye-swap and then combine the results by subtracting the normalized log-ratios:

$$\begin{aligned} M - M' &= \frac{\left(\log_2\left(\frac{R}{G}\right) - c\right) - \left(\log_2\left(\frac{R'}{G'}\right) - c'\right)}{2} \\ &= \frac{\log_2\left(\frac{R}{G}\right) + \log_2\left(\frac{G'}{R'}\right)}{2} \\ &= \frac{\log_2\left(\frac{RG'}{GR'}\right)}{2} , \end{aligned}$$

provided $c = c'$.

Once again, it is worth noting that there is no consensus on the best normalization method. In addition, background correction applies to cDNA arrays as well, and can be implemented by use of limma package.

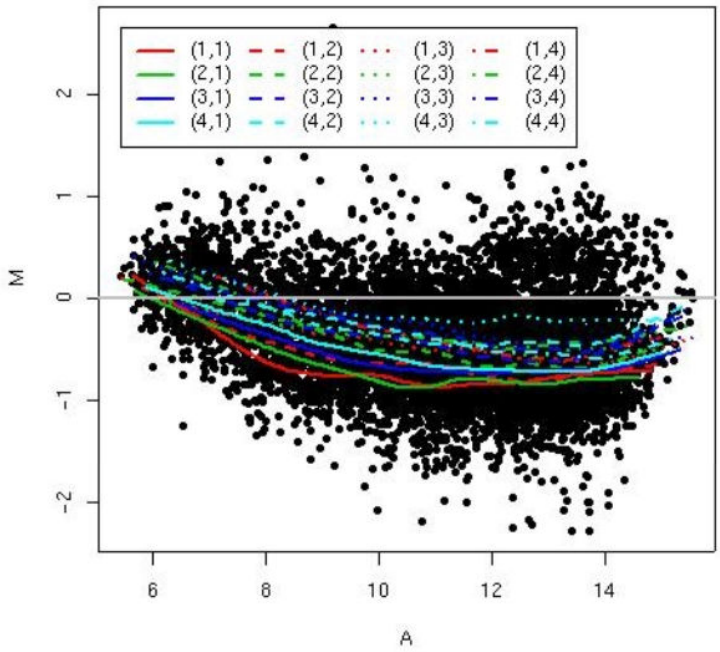


Figure 2.3: MA plot [39]

The results of the above analyses (normalized or raw data) can be made available online on public repositories. Many examples of such data can be downloaded from the Gene Expression Omnibus (GEO) repository [43] or ArrayExpress repository [18].

Table 2.1 summarizes some examples of experiments deposited in the GEO repository. For each study, the series number of the study, its name, the platform used, the normalization performed and the number of columns (samples) is depicted. More details about the prefixes "GSE" and "GPL" and their meaning will be given in section 2.2.

GSE number	Name
GSE3526	Comparison of gene expression profiles across the normal human body
GSE10843	mRNA Cancer Cell Line Profiles
GSE12195	Mutations of multiple genes deregulate the NF-kB pathway in diffuse large B cell lymphoma
GSE5816	A Genome-wide Screen for Hypermethylated Genes in Lung Cancer
GSE36133	Expression data from the Cancer Cell Line Encyclopedia (CCLE)

GSE number	Platform	Number of samples	Normalization
GSE3526	GPL570	353	RMA
GSE10843	GPL570	206	gcRMA
GSE12195	GPL570	136	MAS5.0
GSE5816	GPL570	42	raw
GSE36133	GPL15308	917	RMA

Table 2.1: Several studies from the GEO repository: for each the series number, its name, the platform used, the normalization performed and the number of samples is depicted.

It should be pointed out that in the matrices stored in the repositories, each row corresponds to a probe (or probe set). Then, the standard treatments of annotation (mapping from probes to gene

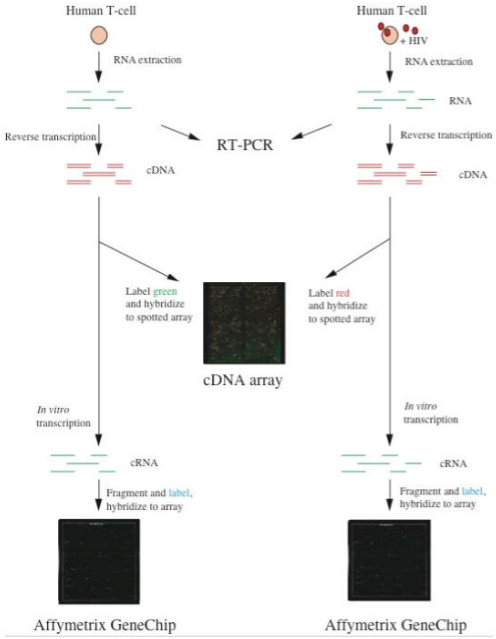


Figure 2.4: Overview of the procedure followed for gene expression measurement with Affymetrix GeneChips and cDNA arrays.

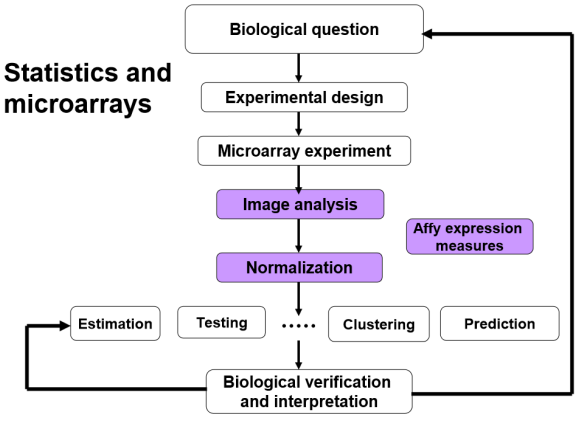


Figure 2.5: Processes taking place before and after the microarray experiment [39]

symbols) and reduction (removal of duplicate gene symbols) need to be performed. These further treatments will be analyzed in section 2.2. In the resulting matrix, each row corresponds to a different gene symbol. We usually refer to both matrices as expression matrices without distinction. However, if there is confusion, we shall be more precise.

Figure 2.4 summarizes the procedure followed in both Affymetrix GeneChips and cDNA arrays, while Figure 2.5 (obtained from a university lecture available from `https://classes.soe.ucsc.edu/bme215/Spring09/PPT/BME215-10-DNAmicroarray.pdf`) presents the flow of processes taking place before and after the microarray experiment.

Finally, some basic differences between Affymetrix and cDNA arrays are listed on Table 2.2.

Affymetrix	cDNA
• They are expensive. • They are commercial arrays. • Short oligonucleotide probes. • Can hybridize only one sample/chip (i.e. no *direct* comparison of two samples). They generate absolute expression values. • Standardized production tends to give good reproducibility.	• They are of lower cost. • They can be customized for each experiment in lab. • Longer probes. • Hybridize two samples/chip (i.e. *direct* comparison of samples). They generate relative expression values. • Non-standardized production can affect reproducibility.

Table 2.2: A comparison between Affymetrix and cDNA arrays.

Another type of microarray widely used is the Illumina BeadChip. The oligonucleotides are longer than in Affymetrix, probes are used instead of probe sets, its construction is based on bead technology and the image file is processed by Bead Studio. More details about Illumina BeadChips can be found in [62].

Another approach for measuring gene expression, that becomes more and more dominant, is the next generation sequencing (NGS). Currently, five NGS platforms are commercially available, including the Roche GS-FLX 454 Genome Sequencer (originally 454 sequencing), the Illumina Genome Analyzer (originally Solexa technology), the ABI SOLiD analyzer, Polonator G.007 and the Helicos HeliScope platforms. The first three are currently dominating the market. Although each NGS platform is unique in how sequencing is accomplished, this approach includes three basic steps: template preparation, sequencing and imaging, and data analysis.

Genomic DNA is fragmented and platform specific "adaptors" are attached. The DNA is then either attached to a bead or directly to the sequencing slide. In either case, the DNA is clonally amplified in this location to provide a cluster of molecules with identical sequences. If beads are used, they are then immobilized on a sequencing slide. Different NGS platforms employ different sequencing chemistries (see web resources). For instance, with the illumina technology, the sequence of each fragment is read by decoding the sequence of fluorophores imaged at each physical position on a sequencing slide. Advanced optics allows for massively parallel sequencing. The procedure followed with the illumina technology is depicted on Figure 2.6. These NGS instruments generate different base read lengths, different error rates, and different error profiles. Once sequencing is complete, raw sequence data must undergo several analysis steps, in order to get an assessment of transcript expression levels. Firstly, the raw reads must be mapped to the reference genome and then the count and summarization of the reads mapped to each known gene, need to take place. This number (usually divided by the transcript length and the total number of reads captured and mapped in the experiment) is the measure of expression level. More specifically, after the experimental procedure, we end up with FASTQ files containing the short reads, together with quality values (raw reads). The reads can be mapped to a reference genome or transcriptome, then summarized at the gene level to produce a matrix of counts with the help of the R/Bioconductor package Rsubread [76]. Other packages can also be used. A more detailed description and specific examples are given in the manuals available at `http://www.bioconductor.org/packages/release/bioc/vignettes/gage/inst/doc/RNA-seqWorkflow.pdf` and `http://www.bioconductor.org/packages/release/bioc/vignettes/limma/inst/doc/usersguide.pdf`.

NGS is gaining more and more ground against microarrays for several reasons. Firstly, microarray technology is limited to detecting transcripts that correspond to existing genomic sequencing information. On the other hand, RNA-seq experiments work well for investigating both known tran-

scripts and exploring new ones. In addition, hybridization issues arisen with microarrays, such as cross-hybridization, do not appear in RNA-seq experiments. Finally, statistical counting of distinct sequences provides a more precise estimate of expression level. However, microarrays still remain popular for two main reasons. Firstly, their longtime use by many researchers has made them an easy to use tool. In addition, despite NGS advancements, expression arrays are still cheaper and easier when a large number of samples is being processed. The present work will mainly focus on gene expression measurements carried out by microarrays. However, the statistical methods described in Chapters 3 and 4 apply to any set of numerical measurements attached to a set of genes, whatever the technology it has been obtained with.

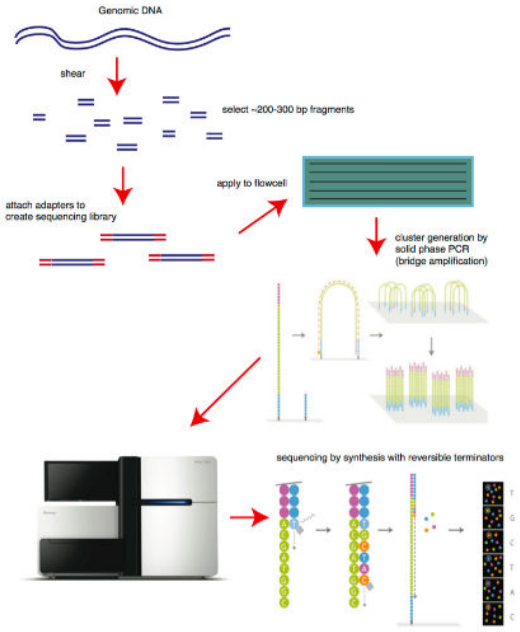


Figure 2.6: NGS with the illumina technology (available at `http://bitesizebio.com/13546/sequencing-by-synthesis-explaining-the-illumina-sequencing-technology/`)

2.1.3 Gene set databases

The other type of statistical variable we shall be working with is the gene set (also named pathway). Of course, it should be noted that a gene set is a much more general and less specific concept than a pathway. However, the two words will be used interchangeably, as the analysis methods are mainly the same. A gene set is a group of genes known to be associated to a given biological process, a particular location in the cell, a molecular function, a type of cancer, etc. The gene sets are grouped together, yielding gene set databases. A very widely used database is the Molecular Signatures Database (MSigDb) [108]. The 7 major collections it consists of, together with a description of the gene sets each one contains, are listed below. The indicated numbers correspond to version 4.0 that has been used in this study.

- C1 (positional gene sets): It contains 326 gene sets. Gene sets corresponding to each human chromosome and each cytogenetic band that has at least one gene. These gene sets are helpful in identifying effects related to chromosomal deletions or amplifications, dosage compensation, epigenetic silencing, and other regional effects.

- C2 (curated gene sets): It contains 4 722 gene sets. Gene sets collected from various sources

such as online gene set databases, publications in PubMed, and knowledge of domain experts. In this collection, we can distinguish several subcollections:

1. CGP (chemical and genetic perturbations): It contains 3 402 gene sets. Gene sets represent expression signatures of genetic and chemical perturbations. A number of these gene sets come in pairs: one contains genes promoted or up-regulated under the perturbation, the other one contains genes repressed or down-regulated.
2. BIOCARTA (BioCarta gene sets): It contains 217 gene sets, derived from the BioCarta gene set database (`http://www.biocarta.com/genes/index.asp`).
3. KEGG (KEGG gene sets): It contains 186 gene sets, derived from the KEGG gene set database (`http://www.genome.jp/kegg/pathway.html`).
4. REACTOME (Reactome gene sets): It contains 674 gene sets, derived from the Reactome gene set database (`http://www.reactome.org/`).

The last three subdatabases can be considered together as one database, the CP (Canonical gene sets).

- C3 (motif gene sets): It contains 836 gene sets. Gene sets that contain genes that share a cis-regulatory motif that is conserved across the human, mouse, rat, and dog genomes. These gene sets make it possible to link changes in a microarray experiment to a conserved, putative cis-regulatory element.
- C4 (computational gene sets): It contains 858 gene sets. Computational gene sets defined by mining large collections of cancer-oriented microarray data.
- C5 (GO gene sets): It contains 1 454 gene sets. Gene sets are named by Gene Ontology (GO) term and contain genes annotated by that term. This collection is divided into three further subcollections:
  1. BP (GO biological process): It contains 825 gene sets, that were derived from the Biological Process Ontology (`http://www.geneontology.org/GO.process.guidelines.shtml`).
  2. CC (GO cellular component): It contains 233 gene sets, that were derived from the Cellular Component Ontology (`http://www.geneontology.org/GO.component.guidelines.shtml`).
  3. MF (GO molecular function): It contains 396 gene sets, that were derived from the Molecular Function Ontology (`http://www.geneontology.org/GO.function.guidelines.shtml`).
- C6 (oncogenic signatures): It contains 189 gene sets. These gene sets represent signatures of cellular gene sets which are often dysregulated in cancer. The majority of signatures were generated directly from microarray data from NCBI GEO or from internal unpublished profiling experiments which involved perturbation of known cancer genes. In addition, a small number of oncogenic signatures were curated from scientific publications.
- C7 (immunologic signatures): It contains 1 910 gene sets. These gene sets represent cell states and perturbations within the immune system. The signatures were generated by manual curation of published studies in human and mouse immunology. For each study, pairwise comparisons of relevant classes were made and genes ranked by mutual information. Gene sets correspond to top or bottom genes (False Discovery Rate < 0.25 or maximum of 200 genes) for each comparison. This resource is generated as part of the Human Immunology Project Consortium (HIPC; `http://www.immuneprofiling.org/`).

Given gene expression matrices and gene set databases, statistical methods aim at identifying genes that are differentially expressed between two or more conditions (e.g. normal and cancer samples, different treatments, good and bad survival) or groups of genes that contain many such genes.

2.2 Data formatting

2.2.1 From raw data to expression matrices

A key element in order to perform any statistical analysis (6-th level of Figure 2.5) are the real data: expression matrices and/or databases. The databases were already in a form ready for use. They can be downloaded from `https://sites.google.com/site/fredsoftwares/products/data-mining` and then used with the help of functions encoded in the R script stdb.r and explained in the manual stdb_manual.pdf (see companion software of [128]). However, the same was not true for the expression matrices, that can be found in abundance, as already stated, in online repositories, such as the GEO repository [43] and the ArrayExpress repository [18].

Our interest focused on the first repository (largest one), which has been developed by the National Center for Bioinformatics (NCBI) at the National Institutes of Health, and is publicly accessible through `http://www.ncbi.nlm.nih.gov/geo`. GEO segregates data into three principal components: platform, sample and series.

- A *platform* is, essentially, a list of probes that define what set of molecules may be detected. Platform accession numbers have a "GPL" prefix.
- A *sample* describes the set of molecules that are being probed and references a single platform used to generate its molecular abundance data. Sample accession numbers have a "GSM" prefix.
- A *series* organizes samples into the meaningful datasets which make up an experiment. Series accession numbers have a "GSE" prefix.

The expression matrices stored on the GEO website need to undergo further treatment in order to obtain a form easy to use by statistical and bioinformatic tools. Our interest focused on single-color arrays. A R function taking as entry a GSE number and downloading the data from the GEO website, formatting them and saving them as two files, has been encoded. There is a recently published R/BioConductor package providing similar tools which perform the above operations: inSilicoDb [111]. A comparison between our R script and the inSilicoDb package will be carried out later. The data are then easy to use to perform any statistical analysis. The function makes use of many R/BioConductor packages. The basic operations which had to be performed (written as subfunctions) were the following:

- Downloading: Before all, the data need to be downloaded from the site. To do so, the existing package to retrieve gene expression data in R, GEOquery [32], was used. We end up with two tables: a table containing the expression levels (data matrix) and one containing the supplementary information for every sample (paired information matrix). More specifically, for the data matrix at this stage, each row corresponds to a different probe (or probe set) identification (id.), each column to a different sample (GSM). The columns of the information matrix are labelled by the same GSM numbers as the paired data matrix. Its rows contain the different information fields of the data.
- Annotation: The data matrices so obtained, have as row names the corresponding probe (or probe set) ids, instead of the gene symbols. Thus, the operation of annotation, that is the mapping from feature names to genomic and functional interpretations, needs to take place. The platforms that can be analyzed by our program are the following: "GPL570", "GPL96", "GPL10558", "GPL6947" and "GPL15308". The mapping between the GPL's and the corresponding platform names is shown on Table 2.3. To perform the annotation, existing R/BioConductor packages were used: hgu133plus2.db [21], hgu133a.db [20], illuminaHumanv3.db [41], illuminaHumanv4.db [42] and org.Hs.eg.db [23]. The first four packages contain annotation data about a particular microarray platform (ChipDb), while the last one contains gene centered data about an organism (OrgDb). All the ".db" packages are updated every six months. In addition, at this stage, the rows corresponding to probe (or probe set) ids with no mapping to a gene symbol are removed.

- Reduction: After performing annotation, some row names may be duplicated, that is some rows may correspond to the same gene symbol. This is due to the fact that most microarrays have multiple probes per gene. Since we must end up with only one row per gene symbol, an operation to remove duplicates must be performed. We chose to give many alternatives to the user. For instance, the row with the maximal interquartile range (IQR) or standard deviation (sd) or any other valid function can be chosen. Probes can be mixed up as well, by taking the mean value, median, maximum, minimum of the duplicated probes for every sample. The user can choose his preferred operation but the proposed one is to perform duplicate removal to retain the highest-IQR probe for each gene (default choice in the R function). After removing duplicates, we end up with a data matrix with each row corresponding to a different gene symbol.

The final data matrix and its paired information matrix are saved as two separate files. The above functions have been encoded as part of a R script, available at `http://ljk.imag.fr/membres/Bernard.Ycart/publis/sagd.tgz` (see [125], subsection 4.6.2 and Appendix). The advantages and disadvantages of our R script and the inSilicoDb package are listed below:

GPL number	Platform name
"GPL570"	Affymetrix Human Genome U133 Plus 2.0 Array
"GPL96"	Affymetrix Human Genome U133A Array
"GPL10558"	Illumina HumanHT-12 V4.0 expression beadchip
"GPL6947"	Illumina HumanHT-12 V3.0 expression beadchip
"GPL15308"	Affymetrix Human Genome U133 Plus 2.0 Array [Brainarray Version 15.0.0, HGU133Plus2_Hs_ENTREZG]

Table 2.3: Mapping between GPL's and platforms.

Our R script

- Advantages

1. The option to choose a method for removing duplicates has been added.
2. Although it handles experiments from (fewer) different platforms, the platform GPL15315, missing from the inSilicoDb package, has been added. In this way, the series GSE36133 [10] (Cancer Cell Line Encyclopedia)- a very important experiment- can be analyzed.
3. Our final objects are files containing matrices. In this way, a user can work with them very easily, without being obliged to know more complex structures (such as expression set object), returned by the other package.

- Disadvantages

1. It takes much more time to obtain the desired matrices, especially when the study contains many samples.
2. It handles fewer platforms than the inSilicoDb package.

inSilicoDb package

- Advantages

1. It is very fast.
2. It handles experiments from many different platforms.
3. The info matrix in some cases is more readable. However, in such cases, people have intervened online in order to give a better form.

- Disadvantages

1. Even for the platforms said to be handled, some experiments may not be able to be analyzed, with the following error message appearing: "Requested study GSExxx on (GPLyyy) is not available."

The basic R function (`read.data`) was used in a study to provide with evidence that reproducible information can be extracted by merging different sources. The study yielded many interesting results (for more information see [125]).

2.2.2 Comparing and merging different studies

As a by-product of the above R function, enabling the choice of the reduction method to be applied, a study was conducted in order to answer the following two questions: 1) For each platform, how many genes had a single probe (or probe set), how many had two, etc.? and 2) Will the adoption of a different method to remove duplicates, influence the results of the statistical analysis to be carried out after?

For the first question, the results are summarized in the tables below:

GPL570

1. Total number of probe sets: 54 675
2. Probe sets mapping to a gene symbol: 41 468
3. Unique gene symbols: 20 108. For these gene symbols we have the following partition (first row: number of probe sets mapping to the same gene symbol, second row: number of gene symbols):

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
9436	5163	2818	1368	706	333	162	58	29	15	7	6	5	1	1

GPL96

1. Total number of probe sets: 22 283
2. Probe sets mapping to a gene symbol: 19 916
3. Unique gene symbols: 12 494. For these gene symbols we have the following partition:

1	2	3	4	5	6	7	8	9	10	11	12	13
8009	2645	1161	434	153	55	17	9	5	2	2	1	1

GPL6947

1. Total number of probes: 49 576
2. Probes mapping to a gene symbol: 29 625
3. Unique gene symbols: 19 404. For these gene symbols we have the following partition:

1	2	3	4	5	6	7	8	9	12	13	20
12851	4020	1773	534	153	44	12	9	4	1	2	1

GPL10558

1. Total number of probes: 47 323
2. Probes mapping to a gene symbol: 34 298
3. Unique gene symbols: 21 187. For these gene symbols we have the following partition:

1	2	3	4	5	6	7	8	9	10
13467	4524	2043	716	245	87	31	23	13	9

11	12	13	14	15	17	19	20	22	24	37
8	5	3	4	1	1	3	1	1	1	1

GPL15308

1. Total number of probe sets: 18 926
2. Probe sets mapping to a gene symbol: 18 615
3. Unique gene symbols: 18 615. In that platform there are no duplicates.

In order to answer the second question, the following approach was adopted: Many GSE studies were chosen. Then, several different methods to remove duplicates were assayed to each one of them: maximal IQR (IQRM), maximal median (medM), maximum applied to columns (max). The minimum IQR (IQRm) and minimum median (medm) were considered as well. Next, in each derived matrix, the actual values were replaced by van der Waerden's normal scores [51, p. 309] and the correlations of the row medians were computed. The results for the study GSE36376 [77] are depicted on Table 2.4 and are typical of what is observed in all cases.

	IQRM	medM	max	IQRm	medm
IQRM	1.00	0.96	0.96	0.74	0.76
medM	0.96	1.00	1.00	0.75	0.72
max	0.96	1.00	1.00	0.75	0.72
IQRm	0.74	0.75	0.75	1.00	0.97
medm	0.76	0.72	0.72	0.97	1.00

Table 2.4: Pairwise correlations of row medians of matrices obtained by different reduction methods.

From Table 2.4, it can be deduced that the similarity of two matrices depends on the reduction method that has been used, if the correlation between the row medians is used as a measure of similarity. For instance, matrices that have been obtained with IQRM, medM or max look quite similar. Also, matrices that have been obtained with IQRm or medm look quite similar too. However, the matrices of the first category are quite different from those of the second.

Another question that arises immediately when such a wealth of data is available, is whether and how these datasets can be compared and/or merged. Several methods addressing the above issue have been proposed: quantile discretization (QD), normal discretization normalization (NorDi), gene quantile normalization (GQ), median rank scores (MRS), the Batch Mean-Centering (BMC) method, Distance-Weighted Discrimination (DWD), Z-score standardization, the Cross-Platform Normalization method (XPN) and an Empirical Bayes (EB) method. The first five methods are implemented in the R/Bioconductor package virtualArray [53], while the rest are implemented in the R/Bioconductor package inSilicoMerging [110]. The EB method can be found in both packages. In addition to these methods, the use of robust statistics has been proposed. Robust methods amount to replacing actual values by ranks, or equivalently by empirical distribution functions or van der Waerden's normal scores [125]. These three choices are implemented as part of a R script available at `http://ljk.imag.fr/membres/Bernard.Ycart/publis/sagd.tgz`.

2.3 Statistical review

In this section, various methods for detecting differentially expressed genes or groups of genes are described. First of all, statistical tests treating each gene separately are presented. Next, statistical tests for gene sets are described. In the second category, the tests are divided into three classes according to their characteristics.

2.3.1 Single-gene (or Gene-wise) analysis

Statistical tests treating each gene separately (single-gene or gene-wise analysis) are considered first. Formally, let x_{ijg} denote the expression level (usually in log scale) of gene g for the i-th sample in condition (group) j. Let us assume first that there are only 2 conditions, N genes in total, $n_1 + n_2$ samples, and set:

$$D_g = \bar{x}_{1g} - \bar{x}_{2g} \quad \text{and} \quad s_g^2 = \frac{(n_1 - 1)s_{1g}^2 + (n_2 - 1)s_{2g}^2}{n_1 + n_2 - 2} \ ,$$

where $\bar{x}_{jg}, (j = 1, 2)$ is the sample mean and s_{jg} the sample standard deviation for group j and gene g (s_g is the pooled sample standard deviation).

The simplest method for identifying differentially expressed genes is the so called "fold" change [99]. This method consists in calculating the ratio of sample means in the case of raw data or their difference in the case of log-transformed data. Thus, in our case the fold change is simply D_g. All genes that differ by more than an arbitrary cut-off value are considered to be differentially expressed. Despite its simplicity, this method is not usually chosen due to the arbitrary nature of the cut-off value and the lack of statistical confidence measures.

Another classical metric to calculate differential expression is the so-called signal-to-noise ratio. It uses the difference between the means, scaled by the standard deviation. It is defined in the following way:

$$\frac{D_g}{s_{1g} + s_{2g}} \ .$$

The larger the signal-to-noise ratio, the more distinct the gene expression is in each group.

The simplest statistically based method for detecting differential expression is the two-sample t-test. The null hypothesis being tested is that there is no difference in expression between the conditions. The test statistic in the ordinary two-sample t-test is:

$$t_g = \frac{D_g}{s_g \sqrt{\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}} \ . \tag{2.3.1}$$

If both samples are normal, the test statistic (2.3.1) follows the Student distribution $St(n_1 + n_2 - 2)$. The standard normal distribution $\mathcal{N}(0, 1)$ can be used for large samples. If we want to test the differential expression between the two conditions, then a two-sided test should be conducted. If the alternative hypothesis is that there is down regulation in the second condition, then a p-value is calculated as the right tail of the theoretical distribution. Finally, if we want to test against overexpression in the second group, the p-value is calculated as the left tail of the theoretical distribution. Genes with p-value below 0.05 are deemed significant. This standard test is usually called gene-specific t-test [31].

However, variance estimates based on a few degrees of freedom can be unreliable. Variances that are severely underestimated can lead to false positives (large value for the t-statistic), while variances that are severely overestimated may lead to a loss of power for detecting differentially expressed genes. Early microarray papers [112, 6] often assumed that variance was constant across all genes. If this assumption held, all the gene specific estimates could be averaged to yield a common estimate of variance for all genes:

$$s^2 = \frac{1}{N} \sum_{i=1}^{N} s_i^2 \ .$$

Such an estimate would have $N(n_1 + n_2 - 2)$ degrees of freedom. In this case, all t-test statistics would have the same denominator:

$$t_g^{gl} = \frac{D_g}{s \sqrt{\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}} \ .$$

For the computation of p-values, the standard normal distribution can be used (as approximation to the Student distribution $St\left(N(n_1 + n_2 - 2)\right)$). We will be referring to this test as global t-test [31].

This is effectively a fold-change test, since the global t-test ranks genes in an order that is the same as fold change. The global t-test may suffer from biases, if the variance is not truly constant for all genes. It is worth noting that this practice is not used much, since the assumption is seldom tenable.

Some other modifications of the t-test have also been proposed with the aim of finding a middle ground in the estimation of variance, that is both powerful and less subject to bias.

In the "significance analysis of microarrays" (SAM) version of the t-test (known as the S test), a small positive constant is added to the denominator of the gene-specific t-test [117]. In this way, genes with small fold change deemed significant only because their estimated variance was small will not be so any more. The test statistic becomes:

$$t_g^{SAM} = \frac{D_g}{s_g\sqrt{\left(\frac{1}{n_1}+\frac{1}{n_2}\right)}+s_0} \, .$$

The term s_0 is a "fudge" factor: its purpose is to prevent the statistic t_g^{SAM} from becoming too large when the variance is close to zero (which can lead to false positives). Tusher et al. [117] initially proposed to determine this factor by a complex procedure based on minimizing the coefficient of variation of t_g^{SAM}. Later, simplified versions have been proposed. For instance, s_0 has been computed as the 90-th percentile of the standard errors of all genes. The significance is estimated through permutation of the group labels.

Another extension of the ordinary t-test is the regularized t-test [8]. The regularized t-test combines information from gene-specific error estimates with a local pooled error estimate based on genes with similar intensities, by using a weighted average of the two as the denominator (Bayesian adjusted denominator) for the t-test. The statistic is:

$$t_g^r = \frac{D_g}{\tilde{s}\sqrt{\left(\frac{1}{n_1}+\frac{1}{n_2}\right)}} \, ,$$

where

$$\tilde{s}^2 = \frac{(n-2)s_g^2 + \nu_0\sigma_0^2}{n-2+\nu_0} \, ,$$

n is the total number of replicates n_1+n_2 per gene across treatments, ν_0 is the degrees of freedom of the prior, and σ_0^2 is the gene specific running average variance. The associated probability of the t-statistic under the null hypothesis is calculated by reference to the Student distribution with $n-2+\nu_0$ degrees of freedom.

Finally, the B statistic proposed by Lönnstedt and Speed [80] is a log posterior odds ratio of differential expression versus non-differential expression. A B statistic of zero corresponds to a $50-50$ chance that the gene is differentially expressed.

The t and B tests can be found in the Statistics for Microarray Analysis (SMA) package. The S test is available in the SAM software package [115] and the regularized t-test is in the Cyber T package [67].

Until now, the assumption that both groups share a common variance has been made. If this is not true (that can be tested by application of the F-test), then the test statistic becomes:

$$t_g = \frac{D_g}{\sqrt{\left(\frac{s_{1g}}{n_1}+\frac{s_{2g}}{n_2}\right)}} \, .$$

The corresponding test is called Welch's t-test. The test statistic follows approximately the Student distribution with

$$\frac{\left(\frac{s_{1g}^2}{n_1}+\frac{s_{2g}^2}{n_2}\right)^2}{\frac{\left(\frac{s_{1g}^2}{n_1}\right)^2}{n_1-1}+\frac{\left(\frac{s_{2g}^2}{n_2}\right)^2}{n_2-1}}$$

degrees of freedom. A p-value is computed from the analytic distribution. If the sample sizes n_1 and n_2 are large, then the distribution can be approximated by a normal $\mathcal{N}(0, 1)$.

In the case the assumption of normality is not met, the non-parametric (robust) equivalent of t-test is the Wilcoxon rank sum test (equivalent to Mann Whitney) (see [51, p. 268-278]).

When three or more conditions have to be distinguished then the one-way anova can be performed if the samples are normally distributed or the non-parametric equivalent Kruskal-Wallis test (see [51, p. 363-372]), in the case the above assumption is violated. In microarray experiments with multiple factors and/or a hierarchy of sources of variation, a mixed anova model or a linear model provide a general and powerful approach to allow full utilization of the information available. With the help of the R/Bioconductor package limma, many such experiments can be analyzed.

The above methods concerned datasets obtained from Affymetrix Genechips. In cDNA microarrays, the situation is a little bit different, since the measurements of the red and green dye on the same array are paired. Let us assume that the logarithms of the expression ratios are given, or more formally that y_{ig} denotes the log expression ratio for array i and gene g. Then, the simplest test to be performed is a one-sample t-test. A detailed presentation of statistical tests for differential expression in cDNA microarray experiments is done in [31].

In the next subsection, the multiple hypothesis testing problem is discussed.

2.3.2 Multiple testing

With a typical microarray dataset comprising thousands of genes, an immediate concern is multiple testing. In multiple testing problems, it is important to control the Type I (false-positive) error rate. The most commonly used approaches control one of the following quantities:

- Family-wise error rate (FWER), that is the probability of at least one Type I error, or else stated of rejecting at least one true null hypothesis.
- False discovery rate (FDR), that is the expected proportion of Type I errors among the rejected hypotheses.

The simplest FWER procedure is the Bonferroni correction: the nominal significance level is divided by the number of tests. Controlling FDR proves more powerful than FWER and is increasingly being adopted in genomic studies. Benjamini and Hochberg [12] provided a multiple testing procedure that guarantees the FDR to be less or equal to a prefixed value q. Benjamini and Yekutieli [13] proposed a simple conservative modification of the original procedure which controls FDR under general dependence. We recommend the Benjamini-Yekutieli method. More details about multiple testing procedures in genomics can be found in [40].

All methods above aim at ranking the genes according to their differential expression. Genes with p-value below 0.05 are declared to be significant. However, the above analysis (at the level of single gene) usually results in very long lists of differentially expressed genes. The biological interpretation of large, "interesting" gene lists (ranging in size from hundreds to thousands of genes) is a challenging and daunting task, dependent on a biologist's area of expertise. Or in some other instances, after correcting for multiple hypothesis testing, no individual gene may be declared significant if the relevant biological differences are modest relative to the noise inherent to the microarray technology. An obvious way to gain biological insight is to assess the differentially expressed genes in terms of their known function(s) or in other words perform gene set (pathway) analysis. By performing analysis at the level of functional group, we can hope to detect modest but coordinate changes of sets of functionally related genes (gene sets of the databases developed in subsection 2.1.3). For instance, an increase of 20% in all genes encoding members of a metabolic pathway may dramatically alter the flux through the pathway and may be more important than a 20-fold increase in a single gene. According to a review of Huang et al. [58], the methods (tools) in functional enrichment analysis can be divided into three classes: singular enrichment analysis (SEA), gene set enrichment analysis (GSEA) and modular enrichment analysis (MEA). The aim of all methods is to reply to some variant of the following question: Are many genes in the gene set differentially expressed (either up or down regulated)?

2.3.3 SEA methods

SEA (also called over-representation analysis (ORA)) is probably the most typical way to make an enrichment analysis. In SEA, on the one hand we have the database and on the other hand we have a vector of genes of interest (user's preselected "interesting" genes). Typically, this vector has arisen from a single-gene analysis of a microarray experiment. The list of all genes is sorted according to their values, that may consist in expression levels, fold-changes, signal-to-noise ratios, t-statistics, correlations, etc. These values may remain unchanged or be transformed. Suitable transformations suggested in the literature include: absolute values, squared values, binary transformation, ranks, p-values or Bayesian posterior probabilities and false discovery rates. There are many instances when a transformation is beneficial, for example in order to improve robustness, or to account for both up- and down-regulation. The binary transformation constitutes an extreme case of ranking, when only two classes (e.g., "differentially expressed" versus "non-differentially expressed") are being considered. Then the first or the last part of the ranked list is being kept (up-regulated genes, down-regulated or both). The number associated to any given gene will be referred to as its *weight*. Denote by N the total number of genes ($N \simeq 20\,000$ for the human genome). Denote by L the set of all genes, by $V \subset L$ the vector of genes of interest and by $G \subset L$ the gene set. Given the vector V, we must decide if its matching or intersection with every gene set in the database is significant or not. The null hypothesis to be tested is the following:

$\mathcal{H}_0$*: The gene sets are random samples without replacement*

Under $\mathcal{H}_0$, the matching follows the hypergeometric distribution. More specifically, let m and n be the number of genes in the vector of interest V and the gene set G respectively. Then, the probability that the two vectors have x genes in common is given by the hypergeometric distribution with parameters m, $N-m$ and n:

$$\text{Prob}(x) = \frac{\binom{m}{x}\binom{N-m}{n-x}}{\binom{N}{n}} = \frac{\binom{n}{x}\binom{N-n}{m-x}}{\binom{N}{m}} \ . \tag{2.3.2}$$

The above test is known as Fisher's exact (FE) test [84, 68]. When N is large enough, the above distribution can be approximated by a binomial distribution with parameters n and $\frac{m}{N}$. If $\frac{nm}{N}$ is also large (in practice we check if $\frac{nm}{N} \geq 5$ and $n\left(1-\frac{m}{N}\right) \geq 5$), the further approximation by a Gaussian distribution $\mathcal{N}\left(\frac{nm}{N}, \frac{nm}{N}\left(1-\frac{m}{N}\right)\right)$ becomes valid.

In case we want to detect significant enrichments of the gene sets (matchings bigger than expected due to chance), then a p-value is computed for every gene set as the right tail probability of the hypergeometric distribution:

$$\text{Prob}(X \geqslant x) = \sum_{i=x}^{\min(m,n)} \frac{\binom{m}{i}\binom{N-m}{n-i}}{\binom{N}{n}} \ . \tag{2.3.3}$$

In case we are interested in depletions (matchings smaller than expected due to chance), then the p-value is computed as the left tail probability. Finally, if both enrichment and depletion are of interest, then a two-sided test should be performed. A more detailed description on the calculation of p-values for the Fisher's exact test can be found in [95]. P-values should be adjusted for multiple hypothesis testing, since many gene sets are tested at the same time. In the literature, there is a confusion between the Fisher's exact test and its approximations. A clarification on that issue is made on the same paper.

Although Fisher's exact test is a very simple test and constitutes a quite efficient way to extract the major biological meaning behind large gene lists, it has been argued by Ycart et al. [128] that the null hypothesis ($\mathcal{H}_0$) assumed, is not really appropriate and may lead to excessive false-positive discoveries. It can be shown that the database vectors are not random samples, since genes are not equally frequent in them but the frequency distribution is clearly right skewed instead. To address this issue, a new testing procedure, called Zelen's exact (ZE) test has been proposed. The new testing method is based on a stratification of genes according to their frequencies. ZE tests the independence of matching numbers conditionally to the strata, measuring significance relatively to the number of

genes in the strata. This solves the false discovery problem and allows more reliable interpretation of lists of genes. More details about the method can be found in [128].

Some publicly available tools implementing Fisher's exact test are: Onto-Tools [69, 37], GOstat[11], GOstats [44], and DAVID [60]. Many of these tools differ very slightly from each other, as they use the same statistical tests as well as overlapping gene set databases. The ZE test has been implemented in the R script stdb.r available at `http://ljk.imag.fr/membres/Bernard.Ycart/publis/stdb.tgz` and in the software tool AutoCompare_ZE-WKS that will be presented in section 4.6.

Although SEA has demonstrated significant success in many genomic studies, the fact that the quality of pre-selected gene lists could largely impact the enrichment analysis, makes it unstable to a certain degree when different cutoff thresholds are used. The actual values of the genes (weights) are not taken into account either. GSEA (also called Functional Class Scoring (FCS)) works in the same way as SEA but taking into account the whole list of genes, with or without their weights, instead of analyzing the most relevant ones. Its main goal is to avoid the arbitrary cutoff.

2.3.4 Detailed description of GSEA

Due to its importance in the present work, this method will be described in greater detail.

Gene Set Enrichment Analysis (GSEA) is a basic tool for genomic data treatment. Since its definition by Subramanian et al. [108], GSEA has been very successful. In Subramanian et al., the weights are ranked in decreasing order. The aim of the method is to know whether the gene set is randomly distributed throughout the ranked list of genes or tends to concentrate at the top or the bottom of it. Or else stated, the question to be answered is: are the weights inside the gene set significantly high or low, compared to weights in a random gene set of the same size?

To answer the question, the following approach is used. Denote by N the total number of genes, by $L = \{g_1, \ldots, g_N\}$ the ranked list of all genes, by $\{w_1, \ldots, w_N\}$ the set of their corresponding weights, by $G \subset L$ the gene set and n its size. In practice, n ranges from a few tens to a few hundreds: n is much smaller than N. GSEA is mainly composed of the following three steps:

- **Calculation of the test statistic**: For each gene set, the distribution of gene ranks from the gene set is compared against the distribution for the rest of the genes, by using a Kolmogorov-Smirnov based test statistic. More specifically, starting from the top of L and walking down the list, a running-sum statistic is increased if the current gene belongs to G and decreased otherwise. The magnitude of the increment depends on the weight of the gene, while the decrement is equal to $\frac{1}{N-n}$. The test statistic, called enrichment score (ES), is defined to be the maximum deviation from zero of this random walk:

$$ES = \max_{i \in \{1,\ldots,N\}} \left| \sum_{\substack{j=1 \\ g_j \in G}}^{i} |w_j| \Big/ \sum_{g_j \in G} |w_j| - \sum_{\substack{j=1 \\ g_j \notin G}}^{i} \frac{1}{N-n} \right| \left(\mathbb{I}_{\max \geqslant - \min} - \mathbb{I}_{\max \leqslant - \min} \right) ,$$

  where $\mathbb{I}$ denotes the indicator of an event and the max and min are calculated over the quantity inside the absolute value.

- **Estimation of significance level of ES**: The statistical significance of the ES (p-value) is estimated through Monte-Carlo simulation. More precisely, the genes or the group labels in the initial experiment are permuted to generate a null distribution of ES. In most cases, the number of permutations is 1 000. The p-value is then calculated relative to this null distribution and in particular by using the positive or negative portion of the distribution corresponding to the sign of the observed ES. As argued by the authors, the permutation of group labels preserves gene-gene correlations, providing thus a more biologically reasonable assessment of significance than would be obtained by gene permutation and as a consequence, group label permutation should be preferred to gene randomization, whenever possible.

- **Adjustment for multiple hypothesis testing**: Since a whole database is tested against the ranked list, the estimated significance level must be adjusted to account for multiple hypothesis

testing. Firstly, the ES for each gene set is normalized to account for the size of the set, yielding a normalized enrichment score (NES). More precisely, the observed ES and the null distribution are normalized, separately rescaling the positive and negative scores by dividing by the mean of the positive and negative portion of the null distribution respectively, to yield the normalized scores. Then, the proportion of false positives is controlled, by calculating the FDR corresponding to each NES, by comparing the tails of the observed and null distributions for the NES.

To facilitate its use, the above methodology has been implemented in a software tool, called GSEA-P [107]. This desktop application is very user-friendly and comes with a detailed manual that can be downloaded from `http://www.broadinstitute.org/gsea/doc/GSEAUserGuideFrame.html`.

In the original implementation of the method by Mootha et al. [89], the running sum statistic used equal weights at every step. In the initial version, the increment was equal to $\frac{1}{n}$. However, if the numeric data are fold changes or correlations, this approach detected as significant gene sets clustered near the middle of the list and thus representing no biological interest. The use of weighting was introduced to address this issue. In the constant weight case, it can be shown that the test statistic, up to a scaling, is the Kolmogorov Smirnov (KS) test statistic for the goodness-of-fit of the uniform distribution on $[0, 1]$ to the sample of the positions of genes of the gene set in the ranked list, once normalized by the population size (see [89], [108, Supporting text, p. 5,6,11], [113], and [128]).

To describe the problem in a more rigorous mathematical setting, it will be convenient to identify the genes to N regularly spaced points on the interval $[0, 1]$, and their absolute weights to the values of a positive valued function g, defined on $[0, 1]$: gene g_i corresponds to point i/N, and its absolute weight $|w_i|$ to $g(i/N)$. The weights appearing to vary smoothly between contiguous genes, the function g can be assumed to be continuous. The gene set can be considered as a subset of size n of the interval $[0, 1]$, say $\{U_1, \ldots, U_n\}$. The absolute test statistic proposed by [108] is then:

$$T_n = \sup_{t\in[0,1]} \left| \frac{\sum_{k=1}^{n} g(U_k)\, \mathbb{I}_{U_k \leqslant t}}{\sum_{k=1}^{n} g(U_k)} - t \right| . \tag{2.3.4}$$

When the weights w_i are constant, then the function g is also constant, and:

$$T_n = \sup_{t\in[0,1]} \left| \sum_{k=1}^{n} \frac{1}{n}\, \mathbb{I}_{U_k \leqslant t} - t \right| . \tag{2.3.5}$$

From (2.3.5), it is obvious that the test statistic T_n is the maximal distance between the empirical CDF of the sample $(U_1, \ldots, U_n)$ and the theoretical CDF of the uniform distribution on the interval $[0, 1]$. In other terms, $\sqrt{n}T_n$ is the Kolmogorov Smirnov (KS) test statistic for the goodness-of-fit of the uniform distribution on $[0, 1]$ to the sample $(U_1, \ldots, U_n)$, as already stated above.

2.3.5 Main flaws of the method

Despite its unquestionable success, GSEA suffers from the following flaws:

1. From a statistical point of view, the centering of its test statistic does not allow the derivation of asymptotic results (see (2.3.4) and subsection 3.1.1).

2. The method ends up being extremely time consuming (in particular, when permutation of sample labels is being performed).

3. Due to the fact that significance is estimated by using the positive or negative portion of the null distribution (see step 2 of GSEA), then although 1 000 permutations are performed in total, the p-value is estimated by less, thus yielding results with very poor precision.

4. The alignment of the null distributions (see step 3 of GSEA) is not carried out properly and this may lead to inaccurate assessment of statistical significance. The division by the mean does not lead to the desired results (accounting for the gene set size).

5. It was mentioned above that the use of weights was motivated to address the issue of gene sets clustered near the middle of the ranked list and thus not exhibiting biological interest but scoring highly though. However, as demonstrated in [36], this aim was not (fully) attained and serious problems remain. GSEA still indicates gene sets with clustering in the middle as statistically significant. For more details and examples of such gene sets, see [36].

6. As will be shown in the next subsection, the null hypothesis assumed by GSEA is competitive, that is all genes are used to calculate the test statistic. However, when group label permutation is performed to estimate the null distribution, the method becomes self-contained (see next subsection). As stated in [114] and [52], this mixing may result in loss of power.

2.3.6 Other FCS methods and comparison

There are some other enrichment tools in the GSEA class using the "no-cutoff" strategy. One of them extends the single-gene analysis method, SAM, to gene set analyses and is called SAM-GS [36]. In SAM-GS, for each of the N genes, the statistic d is calculated in the same way as in SAM for an individual-gene analysis (t^{SAM}). Let $\mathbf{d} = (d_1, \ldots, d_n)$ denote the vector of the test statistics for the genes in the gene set. The SAMGS test statistic is then computed:

$$SAMGS = \sum_{i=1}^{n} d_i^2 \ .$$

A p-value is again estimated through permutations of the group labels (all or a large number of permutations is considered).

A simpler method that has been proposed is the so called parametric analysis of gene set enrichment (PAGE) [70]. PAGE employs fold change between experimental groups or other parametric data to calculate z-scores of predefined gene sets and uses normal distribution to infer statistical significance of gene sets. More specifically, the gene set is identified by the random sample $(X_1, \ldots, X_n)$, with X_i being the value affected to the i-th gene of the gene set. The test statistic employed is:

$$Z_n = \sqrt{n}\left(\frac{\bar{X} - \mu}{\sigma}\right) \ ,$$

where $\bar{X}$ is the sample mean, μ and σ correspond to the mean and standard deviation of the total gene population. If n is large enough, then $Z \sim \mathcal{N}(0, 1)$. A p-value can thus be computed analytically.

In the paper, it is mentioned that $n \geq 10$ suffices for the approximation to be valid. In order to verify their assertion, the following (standard) experiment was conducted: For a given n, a number of gene sets (random samples without replacement from all genes) of size n was simulated, and for each of them the z-statistic was computed. The goodness-of-fit of the (theoretical) normal distribution to the empirical distribution of the sample of simulated values of the test statistic is tested by the (classical) KS test. The values of n range from 5 to 1 000 by step 5. For each n, 10 000 gene sets of size n were simulated. As data, a sample in GSE36133 of Barretina et al. [10], containing expression levels, was used. The results are shown on Figure 2.7. The negative logarithm in base 10 of the KS p-value is plotted. The horizontal line corresponding to p-value 5% has been added. The p-values are small until $n = 40$, they stay above 5% after. The authors state themselves that "the reason that normality analysis of microarray datasets can be performed with a much smaller sampling size (10) than generally required is because parent population of parameters, i.e., fold changes of all genes being compared in microarray datasets, is already somewhat close to normal." However, if the test is to be performed irrespectively of the values (weights), then the approximation might be valid only for larger gene sets ($n \geq 40$).

Another disadvantage of PAGE is that gene sets with bi-directional changes cannot be identified. If a gene set contains both up and down-regulated genes, its score will not be significant, since genes altered in opposite directions will cancel each other. A possible solution to overcome this issue would be to consider the absolute difference $|X_i - \mu|$, instead of the simple difference, in the test statistic. It is worth noting that SAM-GS can also be used to identify gene sets with bi-directional changes.

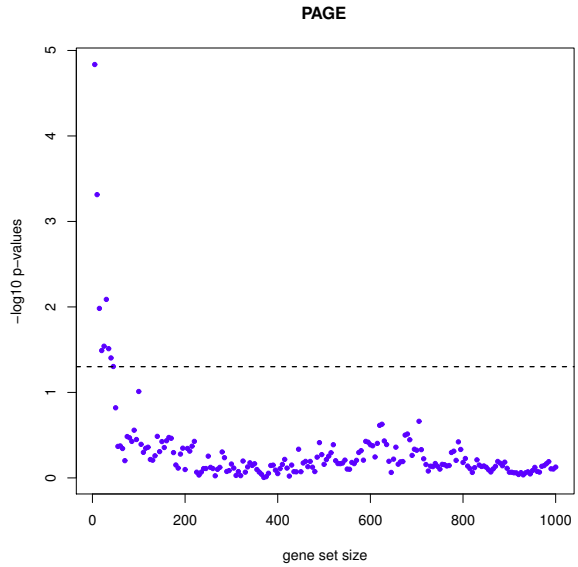


Figure 2.7: Goodness-of-fit of simulated test statistics Z_n over simulated gene sets. The gene set size (abscissa) ranges from 5 to 1 000 by step 5. For each n the ordinate is is the negative logarithm in base 10 of the KS goodness- of-fit p-value over a sample of 10 000 gene sets. The dashed line has ordinate $-\log_{10}(0.05)$.

If gene sets are expected to be interesting due to mean shifts, then it is expected that the z-test outperforms GSEA, since statistical theory predicts this test to be much more powerful than the KS test. Irizarry et al. [64] have also proposed the following test statistic:

$$E_n^z = \sqrt{n}\bar{X}\,,$$

where X_i is the t-statistic of the i-th gene in the gene set. A basic limitation of the above test statistic is that it is only applicable when the summarized values for the genes are t-statistics (or any other statistics which follow a standard normal distribution). The authors state that when enough replicates are available in each condition, then the t-statistics should follow a standard normal distribution under the null hypothesis of no difference between the conditions. Then, the above test statistic follows a standard normal distribution as well. A p-value can thus be easily computed. The authors also mention that a robust version that could be used is the Wilcoxon test. It should be noted that the PAGE method and the z-test of Irizarry et al. are in general different: if in PAGE, the summarized values were t-statistics, then the mean of the gene set would be compared to the mean of all genes, that could happen to be different from zero, if for instance many genes were up-regulated. As already stated, a common limitation of the one-sample z-test is that it will not detect changes in scale. A gene set, where half of the genes are up-regulated and the rest down-regulated may have no mean shift but is certainly interesting from a biological standpoint. To overcome this problem, Irizarry et al. [64] define the following test statistic (standardized χ^2 test statistic) for scale change alternatives:

$$E_n^{\chi^2} = \frac{\sum_{i=1}^n (X_i - \bar{X})^2 - (n-1)}{\sqrt{2(n-1)}}\,.$$

The above test statistic follows the standard normal distribution for n large enough ($n > 20$ according to the authors). Consequently, a p-value can be obtained from this analytical distribution.

In PAGE, due to the fact that the gene set is relatively small with respect to the entire set of all genes, the assumption that genes outside the gene set can be replaced by all genes has been made. However, an ordinary two sample t-test, testing if the mean weights of the genes in the gene set are

significantly different from the mean weights of the genes outside the gene set has also been proposed by Boorsma et al. (see [16]):

$$t^B = \frac{\bar{X}_1 - \bar{X}_2}{s\sqrt{\left(\frac{1}{n} + \frac{1}{N-n}\right)}},$$

where $\bar{X}_1$ are the mean weights of the genes in the gene set, $\bar{X}_2$ are the mean weights of the genes outside the gene set and s is the pooled standard deviation for the two groups: gene set and background set (genes outside the gene set). The associated two-tailed p-value can be calculated from the Student distribution $St(N-2)$. This t-test has been implemented in a tool called T-profiler.

Another known related method is the so-called generally applicable gene set enrichment (GAGE) of Luo et al. [81]. Arguing that the previous two-sample t-test is essentially a one-sample z-test, since the sample size of a gene set is not comparable to its complementary set, the authors propose a different two-sample t-test that accounts for both the gene set specific variance and the background variance as well. The proposed test statistic is the following:

$$t^{GAGE} = \frac{\bar{X}_1 - \bar{X}}{\sqrt{\left(\frac{s_1^2}{n} + \frac{s^2}{n}\right)}},$$

where $\bar{X}_1$ are the mean weights of the genes in the gene set, $\bar{X}$ the mean weights of all genes, s_1^2 the variance of the weights inside the gene set and s^2 the population variance. The above test statistic follows the Student distribution with

$$\frac{\left(\frac{s_1^2}{n} + \frac{s^2}{n}\right)^2}{\frac{\left(\frac{s_1^2}{n}\right)^2}{n-1} + \frac{\left(\frac{s^2}{n}\right)^2}{n-1}},$$

degrees of freedom. A p-value can thus be computed analytically. It should be pointed out that this is a a two sample t-test between the interesting gene set containing n genes and a virtual random set of the same size derived from the background. It should be noted that both GAGE and T-profiler use log-based fold changes as weights. The authors also propose a robust test, equivalent to the Wilcoxon rank sum test.

Finally, Goeman and Bühlmann [52] insist on the distinction between self-contained and competitive hypothesis and propose as a self-contained alternative to GSEA, the application of KS test for evaluating if the p-values of the genes in each gene set are uniformly distributed or not. The competitive methods test the relative enrichment of differentially expressed genes in a gene set compared with the background set, while the self-contained methods use only the information contained in the given gene set. The distinction between the two hypotheses was initiated by [114] and was formally stated in [52]: Let G be the gene set of interest and G^c its complement, then the competitive null hypothesis is

$$\mathcal{H}_0^{comp}: \textit{ The genes in } G \textit{ are at most as often differentially expressed as the genes in } G^c,$$

while the self-contained null hypothesis is

$$\mathcal{H}_0^{self}: \textit{ No genes in } G \textit{ are differentially expressed.}$$

From a practical standpoint, if the null distribution of the test statistic is estimated by permutations, the competitive methods rely on genes as sampling unit and thus can be applied even with one sample per group, yet they cannot work if no genes outside the gene set are measured. On the other hand, the self-contained methods use the subjects as the sampling unit and hence require several samples per group to infer significance of the gene sets. In other words, if no analytical distribution is used, then the null hypothesis and the pursued sampling strategy are interconnected: $\mathcal{H}_0^{comp}$ implies gene sampling, and $\mathcal{H}_0^{self}$ sample label permutation. This "rule" is adopted in most cases, but not always: GSEA does not do so (see explanation next).

Generally, a self-contained test will tend to have more power than a competitive test, as $\mathcal{H}_0^{self}$ tends to imply $\mathcal{H}_0^{comp}$. Many authors, among which Tian et al. [114] and Goeman and Bühlmann [52], favour self-contained tests. The main criticism of self contained tests is that they may be too powerful in settings where many genes appear to be highly differentially expressed: in a situation in which there are many differentially expressed genes almost all gene sets may be called significant.

If we would like to classify the above methods, then SAM-GS and the methods proposed by Irizarry et al. are self-contained methods, PAGE, T-profiler and GAGE are competitive, while GSEA is mixed (both competitive and self-contained): competitive because it uses all genes to calculate the test statistic and self-contained since it uses label permutations to estimate the p-values. It should also be noted that methods whose statistics are based only on G, but use gene randomization to derive the null distribution are competitive (see [114] and [82]).

In FCS methods, there are three ways to calculate the p-values: either use an analytical distribution or estimate the null distribution of the test statistic through permutation of the genes or the group labels. When using permutation of group labels, the gene-gene correlations are preserved. However, the testing procedures doing so, end up being extremely time consuming. In addition, sample randomization requires a certain level of sample replicates to attain deep levels of significance, as already stated, but this condition is not met in many datasets such as time-series data or those designed to investigate the effects of diverse drugs in multiple conditions. Gene permutation takes less time but the assumption that genes be exchangeable must be made. Finally, procedures that end up in analytical distributions make the assumption of independence between genes, an assumption that may not be biologically realistic: as stated in [114], since the genes in the same gene set are functionally related, their expression levels and association metrics are likely to be dependent.

As already stated, GSEA constitutes the most basic tool for genomic data treatment up to date. However, from a statistical point of view, the centering of its test statistic does not allow the derivation of asymptotic results. For this reason, a test statistic with a different centering is presented in Chapter 3. The convergence in distribution of the new test statistic is proved, using the theory of empirical processes. The limiting distribution can be computed by Monte-Carlo simulation. The test defined in this way has been called Weighted Kolmogorov Smirnov (WKS) test. The fact that the evaluation of the asymptotic distribution serves for many different gene sets results in large savings in computing time. Beyond its mathematical and algorithmic advantages, the WKS test could be more informative in many cases, than the classical GSEA test, as will be shown in Chapter 4. A special case of the WKS test arises when the initial weights have been replaced by their ranks. In this case, the estimation of the limiting distribution has been carried out with much greater precision. This makes the WKS test fast and precise, for all uses over rank statistics. We will be referring to this test as WKSr. The WKS test can be either two sided or one sided.

Although the WKS test is a much faster testing procedure than GSEA, it suffers from the same biases as the FE test. Indeed, it will be shown later that for simulated databases where the genes are included with probability proportional to their frequency in the database and simulated lists that tend to rank more frequent genes first (data containing no biological information), an application of the WKS test results in many significant results (false positives). For this reason, an extension of the WKS test, taking into account the unequal frequencies of the genes in the database, will be presented in Chapter 3, as well. We will be referring to this test as Doubly Weighted Kolmogorov Smirnov (DWKS) test. The two tests have been implemented in R scripts that are described in subsection 4.6.3 and are available in the Appendix, but also in the software tool AutoCompare_ZE-WKS, that will be presented in Chapter 4. As already stated, the ZE test has been implemented in the software tool as well and offered as a further option.

In Table 2.3.6, the FCS methods described so far are being summarized. A more detailed reference on the existing methods in this category (FCS methods), together with a classification can be found in [90] or [82]. In addition, a comparison between some methods of this category based on their performance on real or simulated datasets can be found in [78], [35], [105], [2], [122], [48], [61] and [113].

Name (reference)	Test statistic	Tool	Hypothesis	Estimation of the null distribution	Main disadvantage
GSEA [108]	Kolmogorov-Smirnov like test statistic	GSEA-P	competitive	sample (or gene) randomization	time consuming
PAGE [70]	z-statistic	Python script	competitive	analytical N(0,1) distribution	gene independence
method by Irizarry et al. [64]	z-statistic and normalized χ^2		self-contained	analytical N(0,1) distribution	gene independence
method by Broosma et al. [16]	two-sample t-statistic	T-profiler	competitive	Student distribution	gene independence
GAGE [81]	two-sample t-statistic	R package GAGE	competitive	Student distribution	gene independence
SAM-GS [36]	$\sum_{i=1}^{n} d_i^2$	Windows Excel Add-In	self-contained	sample randomization	time consuming
self-contained alternative to GSEA [52]	Kolmogorov-Smirnov test statistic		self-contained	analytical KS distribution or sample randomization	gene independence or time consuming (respectively)
WKS	Weighted cumulated scores	software in Chapter 4	competitive	estimation of analytical distribution	gene independence
WKSr	Weighted cumulated scores	software in Chapter 4	competitive	estimation of analytical distribution	gene independence
DWKS	Weighted cumulated scores	software in Chapter 4 (to be included)	competitive	estimation of analytical distribution	gene independence

Table 2.5: Cut-off free gene set analysis methods. Some basic methods of this category, together with their test statistic, the type of assumed null hypothesis, the way the null distribution is estimated, their main disadvantage and the software tool implementing them are being summarized.

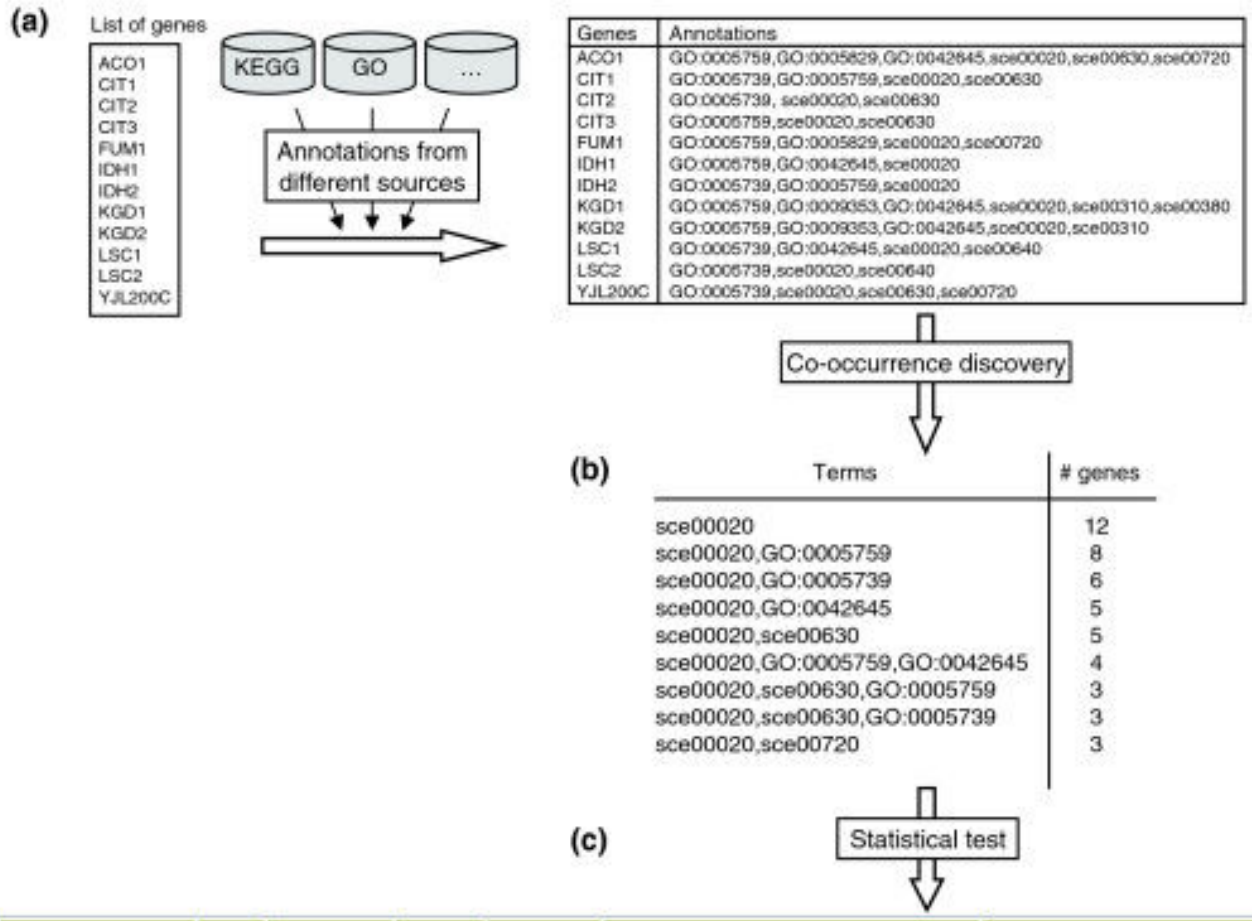
However, tools in the GSEA class are also associated with some common limitations. The requirement of a summarized biological value for each of the genome-wide genes becomes a difficult task when the biological study and genomic platform are complex. Thus, the "no-cutoff" strategy constitutes at the same time the major advantage and basic limitation of GSEA.

2.3.7 MEA methods

Although the above strategies have been demonstrated to be useful, the lack of term-term relationships in their analyses is an important drawback. The combinations of terms may extend our understanding of biological events associated with a given experimental system. MEA inherits the basic enrichment calculation found in SEA but incorporates extra network discovery algorithms by considering the term-term relationships. An example of MEA tool is GeneCodis [24]. It is a method that integrates information of diverse nature by looking for frequent patterns in the space of gene sets (frequent co-occurence of gene sets in a set of genes) and computing their statistical relevance. Figure 2.8 depicts the methodology followed in GeneCodis, applied to a particular example. Two other examples of MEA tools are DAVID [59], that does clustering of singular enrichment results and ProfCom [5], that takes into account different kinds of logic relationships between the gene sets (AND, OR, NOT).

MEA methods take into account the redundant and networked nature of biological annotation content in order to concentrate on building the larger biological picture rather than focusing on an individual gene set or gene. Such data-mining logic seems closer to the nature of biology in that a biological process works in a network manner. However, the quality of pre-selected gene list impacts the analytic result, just as it does in SEA analysis.

More details about the available bioinformatics enrichment tools can be found at [58]. Regardless of their distinct features, these tools have three major layers: data support (backend annotation database), data mining (algorithm and statistics) and result presentation (interface and exploration).



Annotation/s	# List	# Reference	*p*-value	Corrected *p*-value	Genes	Description/s
00020	12(12)	30(6194)	1.90e-28	1.90e-27	CIT2, ACO1, KGD2, LSC2, YJL200C, IDH2, LSC1, KGD1, IDH1, CIT1, FUM1, CIT3	(KEGG)Citrate cycle (TCA cycle)
00020, GO:0005759	8(12)	9(6194)	1.52e-21	1.52e-20	ACO1, KGD2, IDH2, KGD1, IDH1, CIT1, FUM1, CIT3	(KEGG)Citrate cycle (TCA cycle) | (CC)mitochondrial matrix
00020, GO:0005739	6(12)	9(6194)	5.43e-15	5.43e-14	CIT2, LSC2, YJL200C, IDH2, LSC1, CIT1	(KEGG)Citrate cycle (TCA cycle) | (CC)mitochondrion
00020, GO:0042645	5(12)	7(6194)	5.83e-13	5.83e-12	ACO1, KGD2, LSC1, KGD1, IDH1	(KEGG)Citrate cycle (TCA cycle) | (CC)mitochondrial nucleoid
00020, 00630	5(12)	8(6194)	2.62e-12	2.62e-11	CIT2, ACO1, YJL200C, CIT1, CIT3	(KEGG)Citrate cycle (TCA cycle) | (KEGG)Glyoxylate and dicarboxylate metabolism

Figure 2.8: Overview of the methodology in GeneCodis [24]

Chapter 3

The Weighted Kolmogorov Smirnov tests

3.1 Convergence results

First of all, the main convergence results leading to the derivation of the Weighted Kolmogorov Smirnov test [26] and its generalization, Doubly Weighted Kolmogorov Smirnov, are presented. Next, additional mathematical properties concerning the limiting processes are discussed.

3.1.1 Weighted Kolmogorov Smirnov (WKS) testing

A new statistical test was proposed with the aim of fixing the first two limitations of GSEA, as these are reported in subsection 2.3.5.

As already stated in subsection 2.3.4, GSEA aims at comparing a vector of numeric data indexed by the set of all genes, to the genes contained in a given smaller gene set. The numeric data are typically obtained from a microarray experiment. They may consist in expression levels (see [9]), p-values, correlations, fold-changes, t-statistics, signal-to-noise ratios, etc. The number associated to any given gene will be referred to as its *weight*. The gene set may contain genes known to be associated to a given biological process, a cellular component, a type of cancer, etc. Thematic lists of such gene sets are given in the Molecular Signature (MSig) database [108]. The question to be answered is: are the weights inside the gene set significantly different from weights in a random gene set of the same size? As customary in statistical testing, three alternatives to the null hypothesis can be considered, corresponding here to the gene set having significantly higher weights (enrichment), lower weights (depletion) or both (two-tailed test). In what follows, the development of the two-tailed test will be detailed. The theory of the one-tailed case is standard. However, it will be discussed at several places.

Denote by N the total number of genes ($N \simeq 20\,000$ for the human genome). It will be convenient to identify the genes to N regularly spaced points on the interval $[0, 1]$, and their absolute weights to the values of a positive valued function g, defined on $[0, 1]$: gene number i corresponds to point i/N, and its absolute weight $|w_i|$ to $g(i/N)$. In [108], the numbering of the genes is chosen so that weights are ranked in decreasing order. Thus, the weights usually appear to vary smoothly between contiguous genes, and the function g can be assumed to be continuous.

The gene set is included in the set of all genes. Let n be its size. In practice, n ranges from a few tens to a few hundreds: n is much smaller than N. With the identification above, it is considered as a subset of size n of the interval $[0, 1]$, say $\{U_1, \ldots, U_n\}$. If there is no particular relation between the weights and the gene set (null hypothesis), then the gene set must be considered as a uniform random sample without replacement of the set of all genes. The fact that the gene set size n is much smaller than N, justifies identifying the distribution of a uniform n-sample without replacement of $\{1/N, \ldots, N/N\}$ to that of a n-sample of points, uniformly distributed on $[0, 1]$. Therefore, the null hypothesis is:

$\mathcal{H}_0$: *The gene set is a n-tuple $(U_1, \ldots, U_n)$ of independent, identically distributed (i.i.d.) random variables, uniformly distributed on the interval $[0,1]$.*

The basic object is the following step function, cumulating the proportion of weights inside the gene set, along the interval $[0,1]$. It is defined for all t between 0 and 1 by:

$$S_n(t) = \frac{\sum_{k=1}^{n} g(U_k)\, \mathbb{I}_{U_k \leqslant t}}{\sum_{k=1}^{n} g(U_k)} \,. \tag{3.1.1}$$

Recall from (2.3.4) that the absolute GSEA test statistic is:

$$T_n = \sup_{t\in[0,1]} \left| \frac{\sum_{k=1}^{n} g(U_k)\, \mathbb{I}_{U_k \leqslant t}}{\sum_{k=1}^{n} g(U_k)} - t \right| \,.$$

As already stated in subsections 2.3.4 and 2.3.5, our first remark is that in the non constant case, the limit of $S_n(t)$ as n tends to infinity is not t, as (2.3.4) seems to suggest, but instead:

$$\lim_{n\to\infty} S_n(t) = \frac{\int_0^t g(u)\,\mathrm{d}u}{\int_0^1 g(u)\,\mathrm{d}u} \,.$$

Thus the GSEA test statistic T_n in (2.3.4) is not appropriately centered, unless the weights are constant. Instead, the following test statistic should be used:

$$T_n^* = \sqrt{n} \sup_{t\in[0,1]} \left| S_n(t) - \frac{\int_0^t g(u)\,\mathrm{d}u}{\int_0^1 g(u)\,\mathrm{d}u} \right| \,. \tag{3.1.2}$$

Our objective is to derive the asymptotic distribution of T_n^* under the null hypothesis and then deduce from the mathematical result a practical testing procedure. The two-sided case is treated first. The one-sided case, considering the signed difference between $S_n(t)$ and $\int_0^t g(u)\,\mathrm{d}u \Big/ \int_0^1 g(u)\,\mathrm{d}u$ will be analyzed at the end of this subsection.

Our theoretical result is the following.

Theorem 3.1.1. *Let g be a continuous, positive function from $[0,1]$ into $\mathbb{R}$. Denote by G its primitive: $G(t) = \int_0^t g(u)\,\mathrm{d}u$, and assume that $G(1) = 1$. Let $(U_n)_{n\in\mathbb{N}}$ be a sequence of i.i.d. random variables, uniformly distributed on $[0,1]$. For all $n \geqslant 1$, and for all t in $[0,1]$, consider the random variable $S_n(t)$ defined by (3.1.1). Let*

$$Z_n(t) = \sqrt{n}\,(S_n(t) - G(t)) \,. \tag{3.1.3}$$

As n tends to infinity, the stochastic process $\{Z_n(t)\,,\ t \in [0,1]\}$ converges weakly in $\ell^\infty([0,1])$ to the process $\{Z(t)\,,\ t \in [0,1]\}$, where:

$$Z(t) = \int_0^t g(u)\,\mathrm{d}W_u - G(t) \int_0^1 g(u)\,\mathrm{d}W_u \,, \tag{3.1.4}$$

and $\{W_t\,,\ t \in [0,1]\}$ is the standard Brownian motion.

The hypothesis $\int_0^1 g(u)\,\mathrm{d}u = 1$ induces no loss of generality: since g is continuous and positive, its integral is positive; g can be divided by its integral without changing the values of the cumulated proportion of weights $S_n(t)$. The proof of Theorem 3.1.1 is based on the theory of empirical processes, for which [102] and [71] will be used as general references.

The notations of [71] will be used. In particular, throughout the chapter, $\rightsquigarrow$ denotes the weak convergence of processes in $\ell^\infty([0,1])$. The proof of Theorem 3.1.1, which asserts the convergence $Z_n \rightsquigarrow Z$, where Z_n is the empirical process defined by (3.1.3), and Z is the Gaussian bridge defined by (3.1.4) now follows.

Proof. The idea is the following. Consider:

$$Z_n^1(t) = \frac{\sum_{k=1}^n g(U_k)}{n} Z_n(t) \ . \tag{3.1.5}$$

Using the general results on empirical processes and Donsker classes, exposed in section 9.4 of [71], it will be proved that $Z_n^1 \rightsquigarrow Z$. By the law of large numbers,

$$\lim_{n\to\infty} \frac{\sum_{k=}^n g(U_k)}{n} = \int_0^1 g(u)\,\mathrm{d}u = 1 \ , \quad \text{a.s.}$$

The convergence $Z_n \rightsquigarrow Z$ follows as an application of Slutsky's theorem: Theorem 7.15 of [71, p. 112].

The random variable $Z_n^1(t)$ can be written as follows:

$$\begin{aligned} Z_n^1(t) &= \frac{1}{\sqrt{n}} \left(\sum_{k=1}^n g(U_k) \mathbb{I}_{\{U_k \leqslant t\}} - G(t) \sum_{k=1}^n g(U_k) \right) \\ &= \frac{1}{\sqrt{n}} \left(\sum_{k=1}^n g(U_k) \left(\mathbb{I}_{\{U_k \leqslant t\}} - G(t) \right) \right) \ , \end{aligned}$$

denoting by G the primitive of g, as before. Empirical processes are customarily written as function-indexed processes. Define the class of functions $\mathcal{F}$ by:

$$\mathcal{F} = \left\{ \, g(\cdot) \left(\mathbb{I}_{[0,t]}(\cdot) - G(t) \right) \ ; \ t \in [0,1] \, \right\} \ .$$

Denote by $\mathbb{P}_n$ the empirical measure of $(U_1, \ldots, U_n)$, by $\mathbb{P}$ the uniform distribution on $[0,1]$, by $\mathbb{P}_n f$ and $\mathbb{P} f$ the integrals of f with respect to $\mathbb{P}_n$ and $\mathbb{P}$ [71, p. 11]. For $f \in \mathcal{F}$, define $\tilde{Z}_n^1(f)$ by:

$$\tilde{Z}_n^1(f) = \sqrt{n} \left(\mathbb{P}_n f - P f \right) \ . \tag{3.1.6}$$

Obviously, for all $t \in [0,1]$,

$$Z_n^1(t) = \tilde{Z}_n^1 \left(g(\cdot) \left(\mathbb{I}_{[0,t]}(\cdot) - G(t) \right) \right) \ . \tag{3.1.7}$$

Let us prove that $\mathcal{F}$ is a Donsker class. Firstly, observe that the following class $\mathcal{F}_1$ is Donsker.

$$\mathcal{F}_1 = \left\{ \, \mathbb{I}_{[0,t]}(\cdot) - G(t) \, ; \ t \in [0,1] \, \right\} \ .$$

Indeed, for $f \in \mathcal{F}_1$, the process $\sqrt{n} \left(\mathbb{P}_n f - \mathbb{P} f \right)$ converges weakly to the standard Brownian bridge. Since all functions in $\mathcal{F}_1$ take values between -1 and 1, the supremum of $|\mathbb{P} f|$ over $\mathcal{F}_1$ is not larger than 1. The function g, being continuous on a compact interval, is bounded and measurable. From Corollary 9.32, p. 173 of [71], it follows that $\mathcal{F}$ is also Donsker. The convergence of $\tilde{Z}_n^1$ now follows from the result of [71, p. 11]. The limit $\tilde{Z}^1$ is a zero mean, $\mathcal{F}$-indexed, Gaussian process. Its covariance function is defined, for all f_1, f_2 in $\mathcal{F}$ by:

$$\mathbb{E}[\tilde{Z}^1(f_1)\, \tilde{Z}^1(f_2)] = \mathbb{P}(f_1 f_2) - \mathbb{P} f_1\, \mathbb{P} f_2 \ . \tag{3.1.8}$$

Through (3.1.7), the convergence of $\tilde{Z}_n^1$ induces the convergence of Z_n^1, to a zero mean, $[0,1]$-indexed process Z^1. Let us compute the covariance function of Z^1. For s, t in $[0,1]$, let:

$$f_1(\cdot) = g(\cdot)(\mathbb{I}_{[0,s]}(\cdot) - G(s)) \quad \text{and} \quad f_2(\cdot) = g(\cdot)(\mathbb{I}_{[0,t]}(\cdot) - G(t)) \ .$$

Applying (3.1.8) to these functions f_1 and f_2 yields,

$$\begin{aligned} \mathbb{E}[Z^1(t)\, Z^1(s)] &= \int_0^1 g(u)(\mathbb{I}_{[0,s]}(u) - G(s)) g(u)(\mathbb{I}_{[0,t]}(u) - G(t))\,\mathrm{d}u \\ &= \int_0^{\min(t,s)} g^2(u)\,\mathrm{d}u - G(t) \int_0^s g^2(u)\,\mathrm{d}u \\ &\quad -G(s) \int_0^t g^2(u)\,\mathrm{d}u + G(s)G(t) \int_0^1 g^2(u)\,\mathrm{d}u \ . \end{aligned} \tag{3.1.9}$$

There remains to be proved that Z^1 and Z have the same distribution, where Z is defined by the representation (3.1.4) in terms of the standard Brownian motion W:

$$Z(t) = \int_0^t g(u)\,\mathrm{d}W_u - G(t)\int_0^1 g(u)\,\mathrm{d}W_u\ .$$

It is a well known fact that the primitive of a deterministic function with respect to the Brownian motion is Gaussian: therefore Z is a Gaussian process. The covariance function is easily calculated, using formula (32), p. 128 of [102]:

$$\begin{aligned}
\mathrm{Cov}(Z(t),Z(s)) &= \mathrm{Cov}\left(\int_0^t g(u)\,\mathrm{d}W_u - G(t)\int_0^1 g(u)\,\mathrm{d}W_u, \int_0^s g(u)\,\mathrm{d}W_u\right.\\
&- \left. G(s)\int_0^1 g(u)\,\mathrm{d}W_u\right)\\
&= \mathrm{Cov}\left(\int_0^t g(u)\,\mathrm{d}W_u, \int_0^s g(u)\,\mathrm{d}W_u\right)\\
&- G(s)\mathrm{Cov}\left(\int_0^t g(u)\,\mathrm{d}W_u, \int_0^1 g(u)\,\mathrm{d}W_u\right)\\
&- G(t)\mathrm{Cov}\left(\int_0^1 g(u)\,\mathrm{d}W_u, \int_0^s g(u)\,\mathrm{d}W_u\right)\\
&+ G(t)G(s)\mathrm{Cov}\left(\int_0^1 g(u)\,\mathrm{d}W_u, \int_0^1 g(u)\,\mathrm{d}W_u\right)\\
&= \int_0^{\min(t,s)} g^2(u)\,\mathrm{d}u - G(s)\int_0^t g^2(u)\,\mathrm{d}u\\
&- G(t)\int_0^s g^2(u)\,\mathrm{d}u + G(s)G(t)\int_0^1 g^2(u)\,\mathrm{d}u\ .
\end{aligned}$$

It is indeed defined by (3.1.9). The processes Z^1 and Z are both Gaussian, their means and covariance are equal, therefore they have the same distribution. □

Two equivalent representations of the limiting stochastic process defined by (3.1.4) are the following:

$$Z(t) = \int_0^t g(u)\,\mathrm{d}B_u - G(t)\int_0^1 g(u)\,\mathrm{d}B_u\ , \tag{3.1.10}$$

where $\{B_t\,,\ t\in[0,1]\}$ is the standard Brownian bridge and

$$Z(t) = W_{\int_0^t g^2(u)\,\mathrm{d}u} - G(t)W_{\int_0^1 g^2(u)\,\mathrm{d}u}\ . \tag{3.1.11}$$

As explained in the beginning of the subsection, the random variable of interest for GSEA in the two-sided case is the supremum of the process $|Z|$ over the interval $[0,1]$.

Corollary 3.1.2. *Under the notations and hypotheses of Theorem 3.1.1, let*

$$T_n^* = \sup_{t\in[0,1]} |\,Z_n(t)\,|\ .$$

Then T_n^* converges in distribution to

$$\sup_{t\in[0,1]} |Z(t)| = \sup_{t\in[0,1]} \left| \int_0^t g(u)\mathrm{d}W_u - \int_0^t g(u)\mathrm{d}u \int_0^1 g(u)\mathrm{d}W_u \right| ,$$

where W denotes the standard Brownian motion.

Proof. The mapping $f \mapsto \sup_{t\in[0,1]} |f(t)|$, from $l^\infty([0,1])$ into $\mathbb{R}^+$, is continuous. From Theorem 3.1.1, $Z_n \rightsquigarrow Z$. The conclusion follows as an application of Theorem 7.7, p. 109 of [71]. □

As shown in Corollary 3.1.2, the first consequence of Theorem 3.1.1 for GSEA, is that as n increases, the distribution of the proposed test statistic T_n^* under the null hypothesis, tends to that of the following random variable T:

$$T = \sup_{t\in[0,1]} |Z(t)| . \tag{3.1.12}$$

Denote by F its CDF: for all $x > 0$,

$$F(x) = \mathbb{P}[T \leqslant x] . \tag{3.1.13}$$

Observe that $F(x)$ only depends on g, i.e. on the weights of the vector to be tested. Except in the classical KS case of constant weights, F does not have a closed-form expression, but a Monte-Carlo approximation is easily obtained. The testing procedure generalizes that of the classical KS test: since the test statistic T_n^* has asymptotic CDF F under the null hypothesis, the p-value of an observation $T_n^* = x$ is $1 - F(x)$. That testing procedure will be referred to as *Weighted Kolmorov Smirnov* (WKS) test. A crucial feature is that, since F only depends on the weights, the same evaluation of F can be repeatedly used for many gene sets, which saves computing time. Of course, the repeated application of a test to a full database of several thousand gene sets poses the problem of False Discovery Rate (FDR) correction. In applications, we have used the method of [13]: see [40] for multiple testing procedures in genomics and subsection 2.3.2.

Depending on the nature of weights and the purpose of the analysis, the interest might be one or two-tailed (depletion, enrichment or either). Just like the KS test, the WKS was also made one-sided, by testing the signed difference between $S_n(t)$ and $G(t)$. More specifically, if we want to test the null hypothesis that the true distribution function is not greater or less than the hypothesized distribution function, the test statistics employed (under the notations of Theorem 3.1.1) are:

$$T_n^{*g} = \sup(S_n(t) - G(t)) , \tag{3.1.14}$$

and

$$T_n^{*l} = \inf(S_n(t) - G(t)) \tag{3.1.15}$$

respectively. They converge in distribution to the random variables $T^g = \sup_{t\in[0,1]} Z(t)$ and $T^l = \inf_{t\in[0,1]} Z(t)$ respectively (same proof as in Corollary 3.1.2). Denote by F^g and F^l their CDF's respectively. It should be noted that $T^g = -T^l$ in distribution. Indeed,

$$\inf_{t\in[0,1]} Z(t) = - \sup_{t\in[0,1]} (-Z(t)) .$$

However, the stochastic processes Z and $-Z$ are centered Gaussian processes with the same covariance matrices and thus have the same distribution.

A gene set for which $\inf(S_n(t) - G(t))$ is significantly negative, contains genes whose weights tend to be small (down-regulated). Conversely, gene sets for which $\sup(S_n(t) - G(t))$ is significantly positive, contain more up-regulated genes.

3.1.2 Discussion of the WKS test

Accordingly to what has been said in the review section 2.3, the following arguments could be made against WKS:

1. WKS makes the assumption that the random variables are i.i.d., an assumption that is biologically unrealistic. Our counterargument is the one used in all statistical methods making this assumption: dependency (coregulation) is rare for randomly sampled control gene sets. For most curated gene sets there is no coregulation under the specific condition of the microarray study (even though there might be under certain other conditions), and consequently the null hypothesis holds.

2. If the numeric data are fold changes or correlations, then WKS will detect as significant gene sets clustered in the middle of the ranked list, with no biological interest. Our counterargument is that in real applications (with real numeric data and real databases), this behaviour is rarely met, if at all. Very few (if any) gene sets, clustered in the middle, are declared significant. Other solutions could also be proposed. For instance, a transformation could be applied to the numeric data before applying the WKS test.

3. As already remarked in section 2.3, not all genes have the same probability to be included in a database vector. Some of them ("frequent" genes) have higher probability and thus appear in many gene sets, while others have very small probability, thereby being present in only one gene set (see section 4.1). In practice, it is observed that the "frequent" genes tend to rank in the first positions of the list (see section 4.2). This creates the following problem: A large number of gene sets will have many genes in the first part of the ranked list. As a consequence, the cumulated weight function $S_n(\cdot)$ for these gene sets will be biased towards the left: it will be far from G, and the WKS test will return a small p-value, thus detecting the gene set as significant. As a result, a very large number of gene sets is declared to be significant and this is due to the fact that the databases involve unequal gene frequencies. The null hypothesis $\mathcal{H}_0$ appears not to be really appropriate.

3.1.3 Doubly weighted Kolmogorov Smirnov (DWKS) generalization

The generalization of the WKS test to the *Doubly Weighted Kolmogorov Smirnov* (DWKS) test was proposed with the aim of repairing the third limitation of WKS.

Our main theoretical result is summarized in the next theorem.

Theorem 3.1.3. *Let g be a continuous, positive function from $[0,1]$ into $\mathbb{R}$. Let $(X_n)_{n\in\mathbb{N}}$ be a sequence of i.i.d. random variables with space the $[0,1]$ and (common) distribution function F, F being continuous and strictly increasing. Denote by F^{-1} its inverse. For all $n \geqslant 1$ and for all t in $[0,1]$, consider the random variable:*

$$S_n(t) = \frac{\sum_{k=1}^n g(X_k)\mathbb{I}_{X_k\leqslant t}}{\sum_{k=1}^n g(X_k)} \,.$$

Let

$$Z_n(t) = \sqrt{n}\left(S_n(t) - \frac{\int_0^t g(u)\,\mathrm{d}F}{\int_0^1 g(u)\,\mathrm{d}F}\right) = \sqrt{n}\left(S_n(t) - \frac{\int_0^{F(t)} g(F^{-1}(u))\,\mathrm{d}u}{\int_0^1 g(F^{-1}(u))\,\mathrm{d}u}\right) \,.$$

As n tends to infinity, the stochastic process $\{Z_n(t)\,,\ t\in[0,1]\}$ converges weakly in $\ell^\infty([0,1])$ to the process $\{Z(t)\,,\ t\in[0,1]\}$, where:

$$Z(t) = \frac{1}{\int_0^1 g(F^{-1}(u))\,\mathrm{d}u}\left(\int_0^{F(t)} g(F^{-1}(u))\,\mathrm{d}W_u - \frac{\int_0^{F(t)} g(F^{-1}(u))\,\mathrm{d}u}{\int_0^1 g(F^{-1}(u))\,\mathrm{d}u}\int_0^1 g(F^{-1}(u))\,\mathrm{d}W_u\right) \,, \tag{3.1.16}$$

and $\{W_t\,,\ t\in[0,1]\}$ is the standard Brownian motion.

Proof. The random variable $S_n(t)$ can be written in the following way:

$$S_n(t) = \frac{\sum_{k=1}^n g(F^{-1}(F(X_k)))\mathbb{I}_{F(X_k)\leqslant F(t)}}{\sum_{k=1}^n g(F^{-1}(F(X_k)))} \,.$$

But $F(X_k)\,, k = 1, \ldots, n$ are i.i.d. random variables uniformly distributed on $[0, 1]$. Denote by $Z_n^u(t)$ the random variable corresponding to X_i's uniformly distributed on $[0, 1]$. Then $Z_n(t)$ is nothing else but $Z_n^u(F(t))$ for $g = g \circ F^{-1}$. From Theorem 3.1.1, if we replace g by $g \circ F^{-1}$ and t by $F(t)$, we get the desired result. □

Remarks

1. By setting $F^{-1}(u) = r$, the integral $\int_0^{F(t)} g(F^{-1}(u))\,\mathrm{d}u$ can be written in the following way:

$$\begin{aligned}\int_0^{F(t)} g(F^{-1}(u))\,\mathrm{d}u &= \int_0^t g(u)\,\mathrm{d}F \\ &= \int_0^t g(u)f(u)\,\mathrm{d}u\ ,\end{aligned}$$

   where f is the (common) density function, if it exists.

2. If we make the same change of variable, the stochastic integral is written as:

$$\int_0^{F(t)} g(F^{-1}(u))\,\mathrm{d}W_u = \int_0^t g(u)\,\mathrm{d}W_{F(u)}\ .$$

3. The condition in Theorem 3.1.3 that F be inversible has been imposed in order to simplify the proof. However, it suffices that F be continuous for the theorem to be valid. Then the limiting stochastic process Z takes the form inside formula (3.2.34).

The test statistic for DWKS in the two-sided case is the following:

$$T_n^{*D} = \sup_{t\in[0,1]} \left| \sqrt{n} \left(S_n(t) - \frac{\int_0^t g(u)\,\mathrm{d}F}{\int_0^1 g(u)\,\mathrm{d}F} \right) \right|\ .$$

In the next corollary, the asymptotic distribution of T_n^{*D} under the null hypothesis is derived.

The null hypothesis to be tested is:

$\mathcal{H}_0$: *The gene set is a n-tuple $(X_1, \ldots, X_n)$ of independent, identically distributed random variables, with common distribution function F.*

Corollary 3.1.4. *Under the notations and hypotheses of Theorem 3.1.3, let*

$$T_n^{*D} = \sup_{t\in[0,1]} |\, Z_n(t)\,|\ . \tag{3.1.17}$$

Then T_n^{*D} converges in distribution to

$$\sup_{t\in[0,1]} |Z(t)|\ .$$

Proof. The mapping $f \mapsto \sup_{t\in[0,1]} |f(t)|$, from $l^\infty([0, 1])$ into $\mathbb{R}^+$, is continuous. From Theorem 3.1.3, $Z_n \rightsquigarrow Z$. The conclusion follows as an application of Theorem 7.7, p. 109 of [71]. □

The DWKS test has been made one-sided as well just like the WKS and the KS tests. In this case, the signed difference between $S_n(t)$ and $\int_0^t g(u)\,\mathrm{d}F \Big/ \int_0^1 g(u)\,\mathrm{d}F$ is tested. More specifically, if we want to test the null hypothesis that the true distribution function is not greater or less than the hypothesized distribution function F, the test statistics employed (under the notations of Theorem 3.1.3) are:

$$T_n^{*Dg} = \sup \left(S_n(t) - \frac{\int_0^t g(u)\,\mathrm{d}F}{\int_0^1 g(u)\,\mathrm{d}F} \right)\ , \tag{3.1.18}$$

and

$$T_n^{*Dl} = \inf\left(S_n(t) - \frac{\int_0^t g(u)\,\mathrm{d}F}{\int_0^1 g(u)\,\mathrm{d}F}\right) \tag{3.1.19}$$

respectively. They converge in distribution to the random variables $T^{Dg} = \sup_{t\in[0,1]} Z(t)$ and $T^{Dl} = \inf_{t\in[0,1]} Z(t)$ respectively. Denote by F^{Dg} and F^{Dl} their CDF's. It should be noted that $T^{Dg} = -T^{Dl}$ in distribution too.

A gene set for which T_n^{*Dl} is significantly negative, contains more genes whose weights tend to be small (down-regulated) than that of a random sample $(X_1, \ldots, X_n)$ from distribution F (corresponding weights $(g(X_1), \ldots, g(X_n))$). The converse is true for gene sets for which T_n^{*Dg} is significantly positive.

3.1.4 Consistency of (D)WKS test under the alternative

A very important aspect of statistical tests is whether they are consistent under any alternative or not. We will show next that both the WKS and DWKS tests have this desired property.

Lemma 3.1.5. *Let F and F_1 be two continuous distribution functions such that there exists $t \in (0,1)$ such that $|F(t) - F_1(t)| > 0$. Let $(X_n)_{n\in\mathbb{N}}$ be a sequence of i.i.d. random variables with distribution function F_1. Let T_n^{*D} be defined by (3.1.17). Then*

$$\lim_{n\to\infty} T_n^{*D} = +\infty\,, \quad \textit{in probability.}$$

Proof. Let $G(t) = \frac{\int_0^t g(u)\,\mathrm{d}F}{\int_0^1 g(u)\,\mathrm{d}F}$ and $G_1(t) = \frac{\int_0^t g(u)\,\mathrm{d}F_1}{\int_0^1 g(u)\,\mathrm{d}F_1}$. The test statistic becomes:

$$\begin{aligned} T_n^{*D} &= \sqrt{n}\sup_{t\in[0,1]} |\,S_n(t) - G(t)\,| \\ &= \sqrt{n}\sup_{t\in[0,1]} |\,S_n(t) - G_1(t) + G_1(t) - G(t)\,| \\ &\geqslant \sqrt{n}\sup_{t\in[0,1]} |\,G_1(t) - G(t)\,| - \sqrt{n}\sup_{t\in[0,1]} |\,S_n(t) - G_1(t)\,|\,, \end{aligned}$$

which diverges to positive infinity in probability as n tends to infinity, since

$$\sqrt{n}\sup_{t\in[0,1]} |\,S_n(t) - G_1(t)\,| = O_p(1)$$

(bounded in probability) and $\sqrt{n}\sup_{t\in[0,1]} |\,G_1(t) - G(t)\,|$ tends to infinity. □

Remark: The consistency of the WKS test is deduced immediately from the previous lemma, since the WKS test is a special case of the DWKS test, where the distribution F is the uniform distribution.

In the following four subsections, several mathematical results concerning properties of the limiting stochastic process Z and calculations of the limiting distribution in special cases will be derived. For simplicity, the results will concern only the limit process of the WKS test (3.1.4). They can be immediately extended to the DWKS test.

3.1.5 Non-Markovianity

In this subsection, it is proved that the stochastic process Z in (3.1.4) does not have the Markovian property.

Proposition 3.1.6. *Let $Z = \{Z(t)\,,\ t \in [0,1]\}$ be the stochastic process defined in Theorem 3.1.1. Then Z is not Markovian, if $g \neq 1$.*

Proof. Assume that Z is Markovian. Then, since Z is a zero-mean Gaussian process, there should be a non-zero scalar function h_1 and a non-decreasing scalar function h_2 such that for each $t \in [0,1]$ [45]:

$$Z(t) = h_1(t)W(h_2(t)) \ .$$

If we calculate the covariance function of Z for $0 \leqslant s \leqslant t \leqslant 1$, we get:

$$\begin{aligned} \mathrm{Cov}(Z(t), Z(s)) &= \mathbb{E}[Z(t)\, Z(s)] = \mathbb{E}[h_1(t)W(h_2(t))h_1(s)W(h_2(s))] \\ &= h_1(t)h_1(s)\mathbb{E}[W(h_2(t))W(h_2(s))] \\ &= h_1(t)h_1(s)h_2(s) \ . \end{aligned} \tag{3.1.20}$$

From (3.1.20), it follows that there should be a factorization of the covariance function in two functions where one depends only on t and the other only on s. However, in our case, the covariance function, if $0 \leqslant s \leqslant t \leqslant 1$, equals:

$$\int_0^s g^2(u)\,\mathrm{d}u - G(s)\int_0^t g^2(u)\,\mathrm{d}u - G(t)\int_0^s g^2(u)\,\mathrm{d}u + G(s)G(t)\int_0^1 g^2(u)\,\mathrm{d}u \ ,$$

or (if we take out as a common factor $G(s)$)

$$G(s)\left[-\int_0^t g^2(u)\,\mathrm{d}u + G(t)\int_0^1 g^2(u)\,\mathrm{d}u - G(t)\frac{\int_0^s g^2(u)\,\mathrm{d}u}{G(s)} + \frac{\int_0^s g^2(u)\,\mathrm{d}u}{G(s)}\right] \ ,$$

or

$$G(s)\left[-\int_0^t g^2(u)\,\mathrm{d}u + G(t)\int_0^1 g^2(u)\,\mathrm{d}u + \frac{\int_0^s g^2(u)\,\mathrm{d}u}{G(s)}(1 - G(t))\right] \ . \tag{3.1.21}$$

From (3.1.21), we deduce that the covariance function takes the desired form iff

$$\frac{\int_0^s g^2(u)\,\mathrm{d}u}{G(s)} = c \ ,$$

for some constant c. Equivalently,

$$\begin{aligned} \int_0^s g^2(u)\,\mathrm{d}u = c\,G(s) \quad &\Rightarrow \quad g^2(s) = cg(s) \\ &\Rightarrow \quad g(s)[g(s) - c] = 0 \ , \end{aligned}$$

that is, if $g \equiv 0$ or $g \equiv c$ or g is a piecewise constant function with values 0 or c. Thus, we conclude that Z is not Markovian. □

3.1.6 Time reversal property

In this subsection, a property similar to the time reversal property of Brownian bridge is discussed for the limit (3.1.4).

Proposition 3.1.7 (Time reversal)**.** *Under the notations and hypotheses of Theorem 3.1.1, let* $\{Z(t)\,,\ t \in [0,1]\}$ *be the stochastic process corresponding to some function* $g(\cdot)$ *and* $\{Z^*(t)\,,\ t \in [0,1]\}$ *the stochastic process corresponding to the function* $g(1-\cdot)$*. Then the stochastic process* $\{Z^*(t)\,,\ t \in [0,1]\}$ *has the same distribution as* $\{Z(1-t)\,,\ t \in [0,1]\}$*.*

Proof. For the two stochastic processes, the representation (3.1.10) will be used. We have:

$$Z^*(t) = \int_0^t g(1-u)\,\mathrm{d}B_u^1 - \int_0^t g(1-u)\,\mathrm{d}u \int_0^1 g(1-u)\,\mathrm{d}B_u^1 \ ,$$

where B^1 is a standard Brownian bridge. By setting $1-u=r$, we get:

$$\begin{aligned} Z^*(t) &= \int_1^{1-t} g(r)\,\mathrm{d}B_{1-r}^1 + \int_1^{1-t} g(r)\,\mathrm{d}r \int_1^0 g(r)\,\mathrm{d}B_{1-r}^1 \\ &= \int_1^{1-t} g(r)\,\mathrm{d}B_r^2 + \int_1^{1-t} g(r)\,\mathrm{d}r \int_1^0 g(r)\,\mathrm{d}B_r^2 \ , \end{aligned}$$

where B^2 is a standard Brownian bridge ($B_t^2 \equiv B_{1-t}^1$ for $t \in [0,1]$). Next, the term $\int_0^1 g(r)\,\mathrm{d}B_r^2$ being added and subtracted at the same time yields:

$$\begin{aligned} Z^*(t) &= \int_0^1 g(r)\,\mathrm{d}B_r^2 + \int_1^{1-t} g(r)\,\mathrm{d}B_r^2 \\ &- \int_0^1 g(r)\,\mathrm{d}B_r^2 + \int_1^{1-t} g(r)\,\mathrm{d}r \int_1^0 g(r)\,\mathrm{d}B_r^2 \\ &= \int_0^{1-t} g(r)\,\mathrm{d}B_r^2 - \int_0^1 g(r)\,\mathrm{d}B_r^2 - \int_1^{1-t} g(r)\,\mathrm{d}r \int_0^1 g(r)\,\mathrm{d}B_r^2 \\ &= \int_0^{1-t} g(r)\,\mathrm{d}B_r^2 - \int_0^1 g(r)\,\mathrm{d}B_r^2 \left(1 + \int_1^{1-t} g(r)\,\mathrm{d}r\right) \\ &= \int_0^{1-t} g(r)\,\mathrm{d}B_r^2 - \int_0^1 g(r)\,\mathrm{d}B_r^2 \left(\int_0^1 g(r)\,\mathrm{d}r + \int_1^{1-t} g(r)\,\mathrm{d}r\right) \\ &= \int_0^{1-t} g(r)\,\mathrm{d}B_r^2 - \int_0^{1-t} g(r)\,\mathrm{d}r \int_0^1 g(r)\,\mathrm{d}B_r^2 \\ &= \int_0^{1-t} g(u)\,\mathrm{d}B_u^2 - \int_0^{1-t} g(u)\,\mathrm{d}u \int_0^1 g(u)\,\mathrm{d}B_u^2 \\ &= Z(1-t) \quad \text{in distribution} \ , \end{aligned}$$

completing the proof. □

Corollary 3.1.8. *Under the notations and hypotheses of Proposition 3.1.7, the random variables* $\sup_{t\in[0,1]} |Z(t)|$ *and* $\sup_{t\in[0,1]} |Z^*(t)|$ *have the same distribution.*

Proof. Indeed we have:

$$\begin{aligned} \sup_{t\in[0,1]} |Z(t)| &= \sup_{t\in[0,1]} |Z(1-t)| \\ &= \sup_{t\in[0,1]} |Z^*(t)| \quad \text{in distribution} \ , \end{aligned}$$

where the second assertion is a direct consequence of the Proposition 3.1.7. □

Remarks: 1) From assumption $G(1) = \int_0^1 g(u)\,\mathrm{d}u = 1$. Then,

$$\int_0^1 g(1-u)\,\mathrm{d}u = \int_0^1 g(u)\,\mathrm{d}u = 1 \ .$$

2) The same result holds true for the one-sided case as well, that is: the random variables $\sup_{t\in[0,1]} Z(t)$ and $\sup_{t\in[0,1]} Z^*(t)$ have the same distribution.

3.1.7 Explicit calculations

In this subsection, explicit calculations are provided for the limiting cumulative distribution function F (or F^g) defined by (3.1.13), when g is a step function and for the distributions of T_1^*, T_1^{*g} and T_1^{*l}.

Case 1

First of all, the analytical distribution of T will be derived in the special case where g is a step function, satisfying the following relation:

$$g(u)(g(u)-h)=0\,,\quad \text{for } u\in[0,1]\,,$$

where h is a constant in such a way that the constraint $\int_0^1 g(u)\,\mathrm{d}u=1$ be satisfied. In this special case, the following property is true for $u\in[0,1]$:

$$g^2(u)=hg(u)\,, \tag{3.1.22}$$

and thus the covariance function becomes:

$$\begin{aligned}
\mathrm{Cov}(Z(t),Z(s)) &= \int_0^s g^2(u)\mathrm{d}u - G(s)\int_0^t g^2(u)\mathrm{d}u - G(t)\int_0^s g^2(u)\mathrm{d}u \\
&+ G(t)G(s)\int_0^1 g^2(u)\mathrm{d}u \\
&= h\int_0^s g(u)\mathrm{d}u - hG(s)\int_0^t g(u)\mathrm{d}u - hG(t)\int_0^s g(u)\mathrm{d}u \\
&+ hG(s)G(t)\int_0^1 g(u)\mathrm{d}u \\
&= hG(s)-hG(s)G(t) \\
&= hG(s)[1-G(t)]\,.
\end{aligned}$$

But this is the covariance function of the stochastic process $\{\sqrt{h}B_{G(t)} : t\in[0,1]\}$.
Thus, the limiting distribution in these cases can be computed analytically and is equal to:

$$\begin{aligned}
F(t) &= \mathbb{P}[\sup_{t\in[0,1]}|Z(t)|\leqslant x] \\
&= \mathbb{P}[\sup_{t\in[0,1]}|\sqrt{h}B_{G(t)}|\leqslant x] = \mathbb{P}[\sup_{t\in[0,1]}|B_{G(t)}|\leqslant \frac{x}{\sqrt{h}}] \\
&= 1-2\sum_{k=1}^{\infty}(-1)^{k-1}e^{-2k^2x^2/h}\,.
\end{aligned}$$

Another way to obtain the aforementioned result would be to use the representation of Z in (3.1.11):

$$\begin{aligned}
Z(t) &= W_{\int_0^t g^2(u)\,\mathrm{d}u} - G(t)W_{\int_0^1 g^2(u)\,\mathrm{d}u} \\
&= W_{hG(t)} - G(t)W_{hG(1)} \\
&= \sqrt{h}\,\frac{1}{\sqrt{h}}\,W_{hG(t)} - G(t)\sqrt{h}\,\frac{1}{\sqrt{h}}\,W_{hG(1)} \\
&= \sqrt{h}W^1_{G(t)} - G(t)\sqrt{h}W^1_{G(t)} \\
&= \sqrt{h}B_{G(t)}\,,
\end{aligned}$$

where W^1 is another standard Brownian motion ($W^1_t \equiv \frac{1}{\sqrt{h}} W_{ht}$).

The limiting distribution can be deduced very easily for the one-sided case as well:

$$\begin{aligned} F^g(t) &= \mathbb{P}[\sup_{t\in[0,1]} Z(t) \leqslant x] \\ &= \mathbb{P}[\sup_{t\in[0,1]} \sqrt{h} B_{G(t)} \leqslant x] = \mathbb{P}[\sup_{t\in[0,1]} B_{G(t)} \leqslant \frac{x}{\sqrt{h}}] \\ &= 1 - e^{-2\frac{x^2}{h}} \ . \end{aligned}$$

Case 2

Now, the case corresponding to $g(x) = p\mathbb{I}_{[0,a]}(x) + (p-1)\mathbb{I}_{(a,1]}(x)$, where p and a ($0 < a < 1$) are constants will be studied. In this case, the limiting stochastic process Z becomes:

$$Z(t) = p\left(W_t - t\left(W_a + (p-1)W_1\right)\right) \ , \quad \text{for } t \in [0,a]$$

and

$$Z(t) = (p-1)\left(W_t - t\left(W_a + (p-1)W_1\right) + \left(\frac{1-a}{p-1}\right) W_a - aW_1\right) \ , \quad \text{for } t \in (a,1] \ .$$

Thus, for the limiting distribution (without taking into account the absolute value) we have:

$$\begin{aligned} \mathbb{P}[\sup_{t\in[0,1]} Z(t) \leqslant x] &= \mathbb{P}[\sup_{t\in[0,a]} Z(t) \leqslant x, \sup_{t\in[a,1]} Z(t) \leqslant x] \\ &= \mathbb{E}[\,\mathbb{P}[\sup_{t\in[0,a]} Z(t) \leqslant x, \sup_{t\in[a,1]} Z(t) \leqslant x \mid W_a, W_1 - W_a]\,] \ . \end{aligned}$$

But,

1. First of all, the joint distribution of the pair $(W_a, W_1 - W_a)$ is known. More specifically, $W_a \sim N(0,a)$ and $W_1 - W_a \sim N(0, 1-a)$ independently of W_a.

2. $$\begin{aligned} A_1 &= \mathbb{P}[\sup_{t\in[0,a]} Z(t) \leqslant x | W_a = y, W_1 - W_a = z] \\ &= \mathbb{P}[\sup_{t\in[0,a]} p\left(W_t - t\left(W_a + (p-1)W_1\right)\right) \leqslant x | W_a = y, W_1 - W_a = z] \\ &= \mathbb{P}[\sup_{t\in[0,a]} pW_t^{W_a + (p-1)W_1} \leqslant x | W_a = y, W_1 - W_a = z] \\ &= \mathbb{P}[\sup_{t\in[0,a]} pW_t^{y + (p-1)(z+y)} \leqslant x | W_a = y, W_1 - W_a = z] \\ &= \mathbb{P}[\sup_{t\in[0,a]} W_t^{y + (p-1)(z+y)} \leqslant \frac{1}{p}x | W_a = y] \\ &= \mathbb{P}[\sup_{t\in[0,a]} W_t^{y + (p-1)(z+y)} \leqslant \frac{1}{p}x | W_a^{y + (p-1)(z+y)} = (1-ap)y - a(p-1)z] \\ &= \begin{cases} 1 - e^{-2(\frac{1}{p}x)(\frac{1}{p}x - (1-ap)y - a(p-1)z)/a} & , \ \frac{1}{p}x > \max(0, (1-ap)y - a(p-1)z) \\ 0 & , \ \text{otherwise} \end{cases} \ , \end{aligned}$$

   where W^a denotes a standard Brownian motion with drift a and the last result can be found in [17]. Similarly,

3.

$$
\begin{aligned}
A_2 &= \mathbb{P}[\sup_{t\in[a,1]} Z(t) \leqslant x | W_a = y, W_1 - W_a = z] \\
&= \mathbb{P}[\sup_{t\in[a,1]} (p-1)\left(W_t - t\left(W_a + (p-1)W_1\right) + \left(\frac{1-a}{p-1}\right)W_a - aW_1\right) \leqslant x | W_a = y \\
&\quad , W_1 - W_a = z] \\
&= \mathbb{P}[\sup_{t\in[0,1-a]} (p-1)\left(W_{t+a} - (t+a)\left(W_a + (p-1)W_1\right) + \left(\frac{1-a}{p-1}\right)W_a - aW_1\right) \leqslant x \\
&\quad |W_a = y, W_1 - W_a = z] \\
&= \mathbb{P}[\sup_{t\in[0,1-a]} (p-1)\left(W_{t+a} - (t+a)\left(y + (p-1)(y+z)\right) + \left(\frac{1-a}{p-1}\right)y - a(y+z)\right) \leqslant x \\
&\quad |W_a = y, W_1 - W_a = z] \\
&= \mathbb{P}[\sup_{t\in[0,1-a]} (p-1)\left(W_{t+a} - W_a + W_a - (t+a)\left(y + (p-1)(y+z)\right)\right) \leqslant x - \left(\frac{1-a}{p-1}\right)y \\
&\quad + a(y+z) | W_a = y, W_1 - W_a = z] \\
&= \mathbb{P}[\sup_{t\in[0,1-a]} (p-1)\left(W_{t+a} - W_a - (t+a)\left(y + (p-1)(y+z)\right)\right) \leqslant x - \left(\frac{1-a}{p-1}\right)y \\
&\quad + a(y+z) - y | W_a = y, W_1 - W_a = z] \\
&= \mathbb{P}[\sup_{t\in[0,1-a]} (p-1)\left(\tilde{W}_t - (t+a)\left(y + (p-1)(y+z)\right)\right) \leqslant x + \frac{p(a-1)}{p-1}y + az | \tilde{W}_{1-a} = z] \\
&= \mathbb{P}[\sup_{t\in[0,1-a]} (p-1)\tilde{W}_t^{y+(p-1)(y+z)} \leqslant x + \frac{p(a-1)}{p-1}y + az + a(y+(p-1)(y+z)) | \tilde{W}_{1-a} = z] \\
&= \mathbb{P}[\sup_{t\in[0,1-a]} \tilde{W}_t^{py+(p-1)z} \leqslant \frac{1}{p-1}\left(x + \frac{p(ap-1)}{p-1}y + apz\right) | \tilde{W}_{1-a} = z] \\
&= \mathbb{P}[\sup_{t\in[0,1-a]} \tilde{W}_t^{py+(p-1)z} \leqslant \frac{1}{p-1}\left(x + \frac{p(ap-1)}{p-1}y + apz\right) \\
&\quad | \tilde{W}_{1-a}^{py+(p-1)z} = -(1-a)py + z(2-p+a(p-1))] \\
&= \begin{cases} 1 - e^{-2(\frac{1}{p-1}(x+\frac{p(ap-1)}{p-1}y+apz))(\frac{1}{p-1}(x+\frac{p(ap-1)}{p-1}y+apz)-(-(1-a)py+z(2-p+a(p-1))))/(1-a)} \\ 0 \end{cases}, \\
&\quad \begin{cases} \text{if} \quad \frac{1}{p-1}\left(x + \frac{p(ap-1)}{p-1}y + apz\right) > \max(0, -(1-a)py + z(2-p+a(p-1))) \\ \text{otherwise} \end{cases},
\end{aligned}
$$

where $\tilde{W}_t \equiv W_{t+a} - W_a$ and thus $\tilde{W}$ is another standard Brownian motion independent of W_a. Putting the results together, we obtain in total:

$$
\begin{aligned}
\mathbb{P}[\sup_{t\in[0,1]} Z(t) \leqslant x] &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \left(1 - e^{-2(\frac{1}{p}x)(\frac{1}{p}x-(1-ap)y-a(p-1)z)/a}\right) \mathbb{I}_{\{x>\max(0,p(1-ap)y-ap(p-1)z)\}} \\
&\quad \left(1 - e^{-2(\frac{1}{p-1}(x+\frac{p(ap-1)}{p-1}y+apz))(\frac{1}{p-1}(x+\frac{p(ap-1)}{p-1}y+apz)-(-(1-a)py+z(2-p+a(p-1))))/(1-a)}\right) \\
&\quad \mathbb{I}_{\left\{\left(x+\frac{p(ap-1)}{p-1}y+apz\right)>\max(0,-(p-1)(1-a)py+z(p-1)(2-p+a(p-1)))\right\}} \\
&\quad \frac{1}{\sqrt{a}}\frac{1}{\sqrt{1-a}}\phi(\frac{y}{\sqrt{a}})\phi(\frac{z}{\sqrt{1-a}})\mathrm{d}y\mathrm{d}z \ ,
\end{aligned}
$$

where ϕ denotes the standard normal probability density function.

It should be mentioned that the above result generalizes to any step function but the calculations become really complex.

It should be noted that the step functions g, considered in this subsection, do not satisfy the assumption of continuity made in Theorem 3.1.1. The assumption of continuity has been made in order to keep things as simple as possible both for the theorem itself and the further properties. However, the condition that g be continuous can be replaced by the weaker condition that g be continuous almost everywhere and bounded, a condition that is met by the step functions.

Another noteworthy point concerns the fact that the first type step functions g considered, are not positive: they take the value 0 over intervals. Thus, it would be better to consider as test statistic in these cases, the random variable $\sup_{t\in[0,1]} \left| Z_n^1(t) \right|$ defined in the proof of Theorem 3.1.1, while the limiting distribution remains the same.

Case 3

Next, we describe the analytical distributions of T_n^*, T_n^{*g} and T_n^{*l} in the simplest case when $n = 1$.

Lemma 3.1.9. *Let T_1^*, T_1^{*g} and T_1^{*l} be the random variables defined by (3.1.12), (3.1.14) and (3.1.15) respectively. Then*

$$\begin{aligned} \mathbb{P}[T_1^{*g} \leqslant x] &= 1 - G^{-1}(1-x) \text{ , } \textit{for } x \in [0,1] \text{ ,} \\ \mathbb{P}[T_1^{*l} \leqslant x] &= 1 - G^{-1}(-x) \text{ , } \textit{for } x \in [-1,0] \end{aligned}$$

and

$$\mathbb{P}[T_1^* \leqslant x] = G^{-1}(x) - G^{-1}(1-x) \text{ , } \textit{for } x \in [\frac{1}{2}, 1] \text{ .}$$

Proof. If $n = 1$, we have:

$$\sup_{t\in[0,1]} Z_1(t) = 1 - G(U_1) \text{ , } \quad \text{and}$$

$$\inf_{t\in[0,1]} Z_1(t) = -G(U_1) \text{ .}$$

Consequently, we have:

$$\begin{aligned} \mathbb{P}[\sup_{t\in[0,1]} Z_1(t) \leqslant x] &= \mathbb{P}[1 - G(U_1) \leqslant x] = \mathbb{P}[G(U_1) \geqslant 1 - x] \\ &= 1 - \mathbb{P}[G(U_1) < 1 - x] = 1 - \mathbb{P}[U_1 < G^{-1}(1-x)] \\ &= 1 - G^{-1}(1-x) \text{ , for } x \in [0,1] \text{ ,} \end{aligned}$$

$$\begin{aligned} \mathbb{P}[\inf_{t\in[0,1]} Z_1(t) \leqslant x] &= \mathbb{P}[-G(U_1) \leqslant x] = \mathbb{P}[G(U_1) \geqslant -x] \\ &= 1 - \mathbb{P}[G(U_1) < -x] = 1 - \mathbb{P}[U_1 < G^{-1}(-x)] \\ &= 1 - G^{-1}(-x) \text{ , for } x \in [-1,0] \quad \text{and} \end{aligned}$$

$$\begin{aligned} \mathbb{P}[\sup_{t\in[0,1]} |Z_1(t)| \leqslant x] &= \mathbb{P}[\max(1 - G(U_1), G(U_1)) \leqslant x] \\ &= \mathbb{P}[1 - G(U_1) \leqslant x, G(U_1) \leqslant x] \\ &= \mathbb{P}[1 - x \leqslant G(U_1) \leqslant x] \\ &= \mathbb{P}[G^{-1}(1-x) \leqslant U_1 \leqslant G^{-1}(x)] \\ &= G^{-1}(x) - G^{-1}(1-x) \text{ , for } x \in [\frac{1}{2}, 1] \text{ .} \end{aligned}$$

□

3.1.8 Tail estimates

As already stated, the cumulative distribution function F defined by (3.1.13) will be evaluated through Monte-Carlo simulation. However, a drawback of the Monte-Carlo simulation is that it does not accurately estimate low p-values. Showing that a p-value is lower than 10^{-7} for instance, needs at least 10^7 simulations. Next, an upper bound for the right tail of the limiting distribution is derived with the aim of using this for the values for which it constitutes a good approximation. The following lemma describes precisely the result.

Lemma 3.1.10. *Let $Z = \{Z(t)\,,\ t \in [0,1]\}$ be the stochastic process defined in Theorem 3.1.1. Then*

$$\mathbb{P}[\sup_{t\in[0,1]} |Z(t)| > x] \leqslant 2\exp\left[\frac{-x^2}{8\int_0^1 g^2(u)\,\mathrm{d}u}\right] \tag{3.1.23}$$

$$+ \quad 1 - \mathrm{erf}\left(\frac{x}{2\sqrt{2\int_0^1 g^2(u)\,\mathrm{d}u}}\right)\,, \tag{3.1.24}$$

where

$$\mathrm{erf}(\cdot) = \frac{1}{\sqrt{\pi}}\int_{-\cdot}^{\cdot} \exp(-u^2)\,\mathrm{d}u = \frac{2}{\sqrt{\pi}}\int_0^{\cdot} \exp(-u^2)\,\mathrm{d}u\,.$$

Proof. Using the representation of Z in (3.1.11) we have:

$$\begin{aligned}\mathbb{P}[\sup_{t\in[0,1]} |Z(t)| > x] &\leqslant \mathbb{P}[\sup_{t\in[0,1]} |W_{\int_0^t g^2(u)\,\mathrm{d}u}| > \frac{x}{2}] \\ &+ \mathbb{P}[\sup_{t\in[0,1]} |G(t)W_{\int_0^1 g^2(u)\,\mathrm{d}u}| > \frac{x}{2}]\,,\end{aligned} \tag{3.1.25}$$

where a basic inequality has been used. But,

$$\begin{aligned}\mathbb{P}[\sup_{t\in[0,1]} |W_{\int_0^t g^2(u)\,\mathrm{d}u}| > \frac{x}{2}] &= \mathbb{P}[\sup_{t\in\left[0,\int_0^1 g^2(u)\,\mathrm{d}u\right]} |W_t| > \frac{x}{2}] \\ &\leqslant 2\exp\left[\frac{-x^2}{8\int_0^1 g^2(u)\,\mathrm{d}u}\right]\,,\end{aligned} \tag{3.1.26}$$

where the basic Doob's inequality for the standard Brownian motion has been applied and

$$\begin{aligned}\mathbb{P}[\sup_{t\in[0,1]} |G(t)W_{\int_0^1 g^2(u)\,\mathrm{d}u}| > \frac{x}{2}] &= \mathbb{P}[|W_{\int_0^1 g^2(u)\,\mathrm{d}u}| \sup_{t\in[0,1]} |G(t)| > \frac{x}{2}] \\ &= \mathbb{P}[|W_{\int_0^1 g^2(u)\,\mathrm{d}u}| > \frac{x}{2}] \\ &= 1 - \mathrm{erf}\left(\frac{x}{2\sqrt{2\int_0^1 g^2(u)\,\mathrm{d}u}}\right)\,.\end{aligned} \tag{3.1.27}$$

By replacing (3.1.26) and (3.1.27) in (3.1.25), we get:

$$\mathbb{P}[\sup_{t\in[0,1]}|Z(t)|>x] \leqslant 2\exp\left[\frac{-x^2}{8\int_0^1 g^2(u)\,\mathrm{d}u}\right] \tag{3.1.28}$$

$$+ \quad 1-\mathrm{erf}\left(\frac{x}{2\sqrt{2\int_0^1 g^2(u)\,\mathrm{d}u}}\right) . \tag{3.1.29}$$

□

For the one-tailed case, the following bound is obtained similarly:

$$\mathbb{P}[\sup_{t\in[0,1]}Z(t)>x] \leqslant \exp\left[\frac{-x^2}{8\int_0^1 g^2(u)\,\mathrm{d}u}\right] \tag{3.1.30}$$

$$+ \quad \frac{1}{2}\left(1-\mathrm{erf}\left(\frac{x}{2\sqrt{2\int_0^1 g^2(u)\,\mathrm{d}u}}\right)\right) . \tag{3.1.31}$$

3.2 Implementation

In this section, the implementation of the two-tailed WKS and DWKS test is described in detail. The one-sided versions are presented too.

3.2.1 The WKS test

Several issues regarding the implementation of the WKS test are discussed here. The essential step is the evaluation of the cumulative distribution function F defined by (3.1.13), or else:

$$F(x)=\mathbb{P}\left[\sup_{t\in[0,1]}\left|\int_0^t g(u)\,\mathrm{d}W_u - G(t)\int_0^1 g(u)\,\mathrm{d}W_u\right|\leqslant x\right] . \tag{3.2.32}$$

A Monte-Carlo calculation has to be used. First of all, sample paths for the stochastic process

$$\left\{\int_0^t g(u)\,\mathrm{d}W_u\,;\ t\in[0,1]\right\}$$

must be simulated. This is done using a standard Euler-Maruyama scheme: see [97] for a review of numerical methods for stochastic integrals and differential equations. A regular subdivision of the interval $[0,1]$ into m intervals is chosen:

$$t_i=\frac{i}{m}\,,\ i=0,\ldots,m\ .$$

Recall that in practice, the function g is known at points i/N representing the genes. Hence it is natural to choose $m=N$. The stochastic integral is approximated by a sum:

$$\int_0^t g(u)\,\mathrm{d}W_u \approx \sum_{i=0}^{m-1} g(t_i)\,(W_{t_{i+1}\wedge t}-W_{t_i\wedge t})\ . \tag{3.2.33}$$

The increments $W_{t_{i+1}}-W_{t_i}$ are easily simulated as i.i.d. centered Gaussian variables, with variance $1/m$. An estimate of the CDF F is obtained by simulating $nsim$ discretized trajectories of Z, taking the maximum of the absolute value of each, then returning the empirical CDF of the obtained sample. The algorithm can be written as follows.

Algorithm 1 Approximation of F

1: Simulate increments of the Brownian motion on $t_0, \ldots, t_m$,
2: for $i = 0, \ldots, m-1$, compute $g(t_i)\,(W_{t_{i+1}} - W_{t_i})$,
3: get cumulated sums of the previous sequence,
4: deduce the discretized trajectory for $\{Z(t)\,,\ t \in [0,1]\}$ at $t_0, \ldots, t_m$,
5: compute the maximum absolute value of the previous sequence,
6: repeat $nsim$ times steps 1 to 5,
7: return the empirical distribution function of the obtained sample.

Actually, since $F(x)$ is evaluated as the proportion of a sample below x, the result must take the uncertainty into account. We propose to return the lower bound of the 95% left-sided confidence interval, instead of the point estimate. This gives an upper bound for the p-value, which is a conservative evaluation. As stated before, the CDF F only depends on the weight function g. The relation between g and F is illustrated on Figure 3.1. Five different CDF's have been computed, for $g_k(x) = (k+1)(1-x^{1/k})$, $k = 0, 1, 2, 3, 4$. Denote them by $F_0, \ldots, F_4$. The case $k = 0$ is that of constant weights, and can be used as a validation for the algorithm above: F_0 is the Kolmogorov Smirnov CDF, which has an explicit expression. It can be checked that the estimate output by Algorithm 1 is close to the known exact function (see also subsection 3.3.2). The curves of Figure 3.1 were obtained via 20 000 Monte-Carlo simulations, over 15 000 discretization points. It turns out that for all x, $F_0(x) > \cdots > F_4(x)$: the steeper g, the smaller F, and the larger the p-values. The differences between the curves are sizable: calculating $\sup |F_k - F_0|$ for $k = 1, \ldots, 4$ gives 0.199, 0.271, 0.324, 0.356. Theoretical functions g may seem of little practical interest. This is not so, for two reasons. The

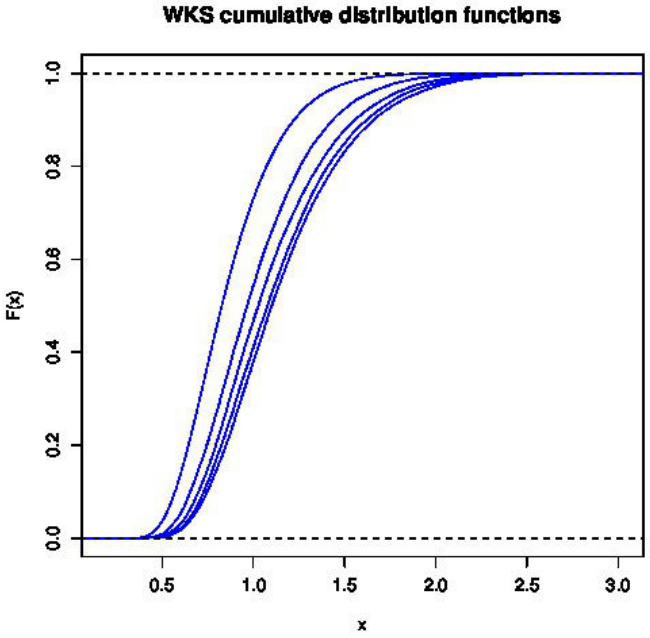


Figure 3.1: Cumulated distribution functions F_k corresponding to $g_k(x) = (k+1)(1-x^{1/k})$, for $k = 0, 1, 2, 3, 4$. The highest curve corresponds to $k = 0$ (constant weights, classical Kolmogorov Smirnov CDF). The CDF's decrease as k increases: the steeper g, the smaller F, and the larger the p-values.

first reason is the use of robust statistics (see [54] as a general reference, and [116] for application to expression data). If the initial values are replaced by their ranks, then the weights are $N, N–1, \ldots, 2, 1$.

Therefore, the weight function is $g_1(x) = 2(1-x)$. This justifies calculating F_1 with good precision, which makes the WKS test fast and precise, for all uses over rank statistics. We have done so, using 10^6 Monte-Carlo simulations, and 10^5 discretization points. The second reason is the observation of F when the weights come from real data. Eight different GEO datasets were considered: GSE36382 [86], GSE48348 [119], GSE36809 [124], GSE31312 [47], GSE48762 [91], GSE37069 [100], GSE39582 [85], and GSE9984 [87]. Several samples of expression levels in each study were selected. In each sample, the expression levels were ranked in decreasing order, and Algorithm 1 was applied in order to obtain an estimation of F. For all real datasets, the estimated F was such that $F_4(x) < F(x) < F_0(x)$ (see Figure 3.2). It seems to be the case in practice that F_4 and F_0 provide lower and upper bounds for F.

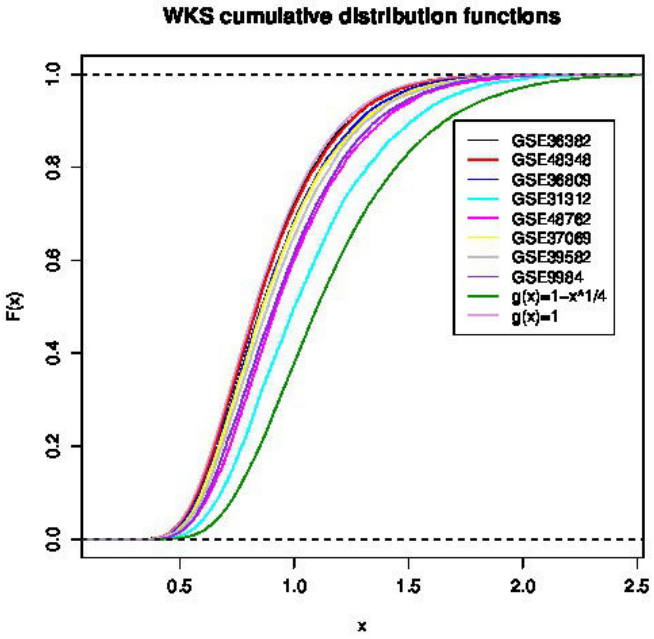


Figure 3.2: Cumulated distribution functions F corresponding to weights coming from real data. Each distribution corresponds to a sample of expression levels selected at random from the following eight GEO datasets: GSE36382, GSE48348, GSE36809, GSE31312, GSE48762, GSE37069, GSE39582 and GSE9984. The distributions F_0 and F_4, obtained before, have been added to the plot.

The next algorithmic point concerns the calculation of the test statistic, that is the value of T_n^* defined by (3.1.2) for a given set of weights and a gene set of size n:

$$T_n^* = \sqrt{n} \sup_{t\in[0,1]} |\, S_n(t) - G(t) \,| \ ,$$

where

$$S_n(t) = \frac{\sum_{k=1}^n g(U_k)\, \mathbb{I}_{U_k \leqslant t}}{\sum_{k=1}^n g(U_k)} \ .$$

The values $g(U_k)$ are the weights of genes inside the gene set. Observe that, if the same vector has to be tested against many gene sets, the calculation of $G(t)$ (cumulated sums of all weights) must be done only once. The value of T_n^* is returned by a procedure similar to that of the classical KS test. Consider two non-decreasing functions f and h where f is a step function with jumps on the set $\{x_1, \ldots, x_n\}$ and h is continuous. The supremum of the difference between f and h is computed as follows [7, p. 35].

$$\sup_x |\, f(x) - h(x) \,| = \max_i \{\, \max\{\, |\, h(x_i) - f(x_i) \,|, |\, h(x_i) - f(x_{i-1}) \,|\,\}\,\} \ .$$

3.2.2 The DWKS test

The essential step for the implementation of the DWKS test is the evaluation of the cumulative distribution function F^D defined by:

$$F^D(x) = \mathbb{P}\left[\sup_{t\in[0,1]} \left| \frac{1}{\int_0^1 g(u)\,\mathrm{d}F} \left(\int_0^t g(u)\,\mathrm{d}W_{F(u)} - \frac{\int_0^t g(u)\,\mathrm{d}F}{\int_0^1 g(u)\,\mathrm{d}F} \int_0^1 g(u)\,\mathrm{d}W_{F(u)} \right) \right| \leqslant x \right] . \quad (3.2.34)$$

For the evaluation of the limiting distribution, the procedure described in subsection 3.2.1 has been followed. For the simulation of sample paths of the limiting stochastic process Z in (3.1.16), the following approximations were used:

$$\int_0^t g(u)\,\mathrm{d}F \approx \sum_{i=0}^{m-1} g(t_i)\,(F_{t_{i+1}\wedge t} - F_{t_i\wedge t})\ ,$$

and

$$\int_0^t g(u)\,\mathrm{d}W_{F(u)} \approx \sum_{i=0}^{m-1} g(t_i)\,(W_{F_{t_{i+1}\wedge t}} - W_{F_{t_i\wedge t}})\ ,$$

where a regular subdivision of the interval $[0,1]$ into m intervals is chosen:

$$t_i = \frac{i}{m}\,,\ i = 0,\ldots,m\ .$$

The increments $W_{F_{t_{i+1}}} - W_{F_{t_i}}$ are easily simulated as i.i.d. centered Gaussian variables, with variance $F_{t_{i+1}} - F_{t_i}$:

$$W_{F_{t_{i+1}}} - W_{F_{t_i}} = \sqrt{F_{t_{i+1}} - F_{t_i}}N(0,1)\ .$$

An estimate of the CDF F^D is obtained by simulating $nsim$ discretized trajectories of Z in (3.1.16), taking the maximum of the absolute value of each, then returning the empirical CDF of the obtained sample. The algorithm can be written as follows.

Algorithm 2 Approximation of F^D

1: Simulate increments of the Brownian motion on $F(t_0),\ldots,F(t_m)$,
2: for $i = 0,\ldots,m-1$, compute $g(t_i)\,(W_{F_{t_{i+1}}} - W_{F_{t_i}})$,
3: get cumulated sums of the previous sequence,
4: deduce the discretized trajectory for $\{Z(t)\,,\ t\in[0,1]\}$ at $t_0,\ldots,t_m$,
5: compute the maximum absolute value of the previous sequence,
6: repeat $nsim$ times steps 1 to 5,
7: return the empirical distribution function of the obtained sample.

In addition, it is worth noting that when $g(x) = 1 - F(x)$, then $g(F^{-1}(x)) = 1 - x$ and thus the cumulative distribution function F_1 that has been calculated with very good precision (see subsection 3.2.1) can be used.

The value of the test statistic T_n^{*D} will be computed in the same way as for the WKS test.

3.2.3 One-sided versions

The same details will be briefly discussed in the case of the one-sided WKS test. The cumulative distribution function F^g of T^g defined by:

$$F^g(x) = \mathbb{P}\left[\sup_{t\in[0,1]} \left(\int_0^t g(u)\,\mathrm{d}W_u - G(t)\int_0^1 g(u)\,\mathrm{d}W_u \right) \leqslant x \right] \quad (3.2.35)$$

was evaluated by use of Algorithm 1. The only difference consists in computing the maximum value of the sequence in step 5, in the place of the maximum absolute value. As far as the cumulative distribution function F^l of T^l, its relation to F^g was used. Regarding the values of T_n^{*g} and T_n^{*l}, they will be computed using the following relations:

$$\sup_x(\, f(x) - h(x)\,) = \max_i\{\, f(x_i) - h(x_i)\,\}$$

and

$$-\inf_x(\, f(x) - h(x)\,) = \max_i\{\, h(x_i) - f(x_{i-1})\,\}\ .$$

The same precise calculation of F_1^g (and thus of $1 - F_1^l$), corresponding to $g_1(x) = 2(1-x)$, was carried out as well, using 10^6 Monte-Carlo simulations, and 10^5 discretization points. The two limiting distribution functions for the two-sided and one-sided WKS test, that will be used over rank statistics, are depicted together on Figure 3.3.

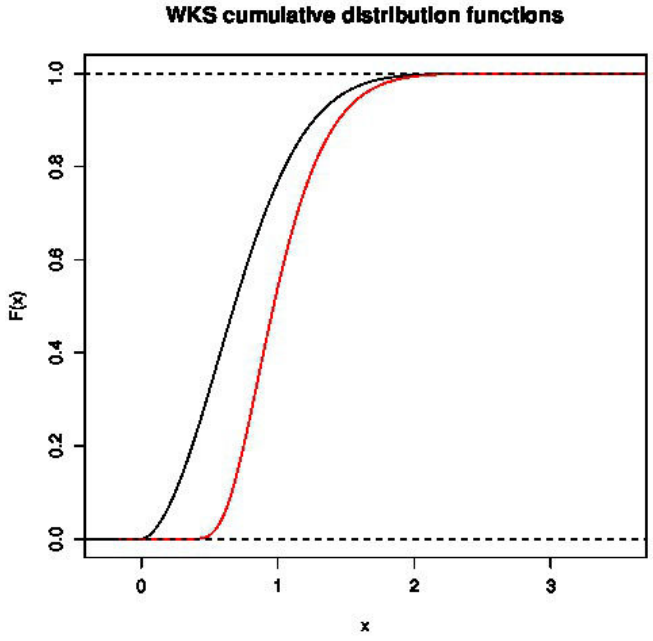


Figure 3.3: Cumulated distribution functions F_1 and F_1^g corresponding to $g_1(x) = 2(1-x)$. The red curve corresponds to the two-sided WKS test, while the black is used for the one-sided WKS test.

From Figure 3.3, it is clear that the cumulative distribution function of T is below that of T^g, as expected, since $T \geqslant T^g$. In addition, it is obvious that the difference between the two curves decreases as x increases. This is due to the fact that the random variables T^g and T^l are negatively correlated.

In order to explore the relation (dependency or not) between the random variables T^g and T^l, 20 000 values of the pair $\left(T^g, -T^l\right)$ were simulated. The result is shown on Figure 3.4. As expected, the two random variables are negatively correlated.

The implementation of the one-sided DWKS test can be easily deduced accordingly.

3.2.4 User choices

The distribution function F is passed as an argument to the R function implementing the DWKS test. A reasonable and simple choice for F will be presented. The model proposed in [128] will be adopted. Assume a partition $\{T_1, \ldots, T_k\}$ of the set of N genes is given (with k much smaller than N). Each

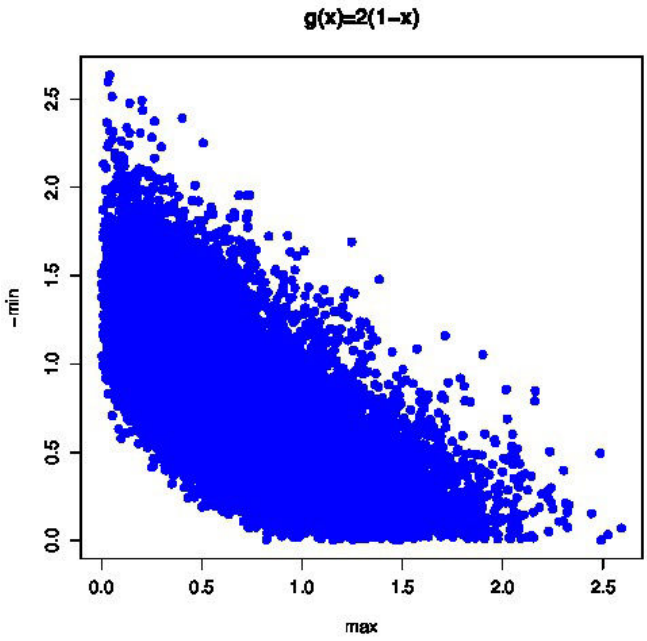


Figure 3.4: Scatterplot of the pair of random variables $\left(T^g, -T^l\right)$, for $g(x) = 2(1-x)$.

gene j belongs to one and only one set T_j, that will be referred to as its type. We denote by N_j the number of genes of type T_j and assume that the inclusion probabilities (probability of a gene to be included in the gene set) are constant over types. Therefore, there exist only k different probabilities, that will be denoted by $\pi_1, \ldots, \pi_k$, π_j being the common inclusion probability of all genes of type T_j (given or estimated). Thus, here we make the assumption that the gene sets are no more random samples without replacement from the gene population but that we have k groups of genes and that the gene sets are samples without replacement from the gene population where the genes are included according to their inclusion probability. Although it will not be further used due to a simplification that will be made, it should be noted that in our initial hypothesis the samples are drawn according to the Wallenius model [46]. Then we have:

$$\mathbb{P}[X_1 \leqslant t] = \sum_{i=1}^{k} \mathbb{P}[X_1 \leqslant t | \text{gene is of type } T_i] \mathbb{P}[\text{gene is of type } T_i] \; . \tag{3.2.36}$$

But,

$$\mathbb{P}[\text{gene is of type } T_i] = N_i \pi_i \; , \quad \text{for } i = 1, \ldots, k \tag{3.2.37}$$

and the conditional distribution functions $\mathbb{P}[X_1 \leqslant t | \text{gene is of type } T_i]$ will be estimated through the following relation for $i = 1, \ldots, k$:

$$\begin{aligned} \mathbb{P}[X_1 \leqslant t | \text{gene is of type } T_i] &\approx \frac{\#\{\text{genes of type } T_i \text{ until position } t\}}{\#\text{genes of type } T_i} \\ &= \frac{\#\{\text{genes of type } T_i \text{ until position } t\}}{N_i} \; . \end{aligned} \tag{3.2.38}$$

If we replace (3.2.37) and (3.2.38) in (3.2.36), we get:

$$\begin{aligned}\mathbb{P}[X_1 \leqslant t] &\approx \sum_{i=1}^{k} \frac{\#\{\text{genes of type } T_i \text{ until position } t\}}{N_i} N_i \pi_i \\ &= \sum_{i=1}^{k} \#\{\text{genes of type } T_i \text{ until position } t\}\pi_i \,.\end{aligned}$$

The inclusion probabilities π_j will be estimated in the same way as in [128]. More specifically, the estimation will be based on the gene frequencies in the database. The distribution of gene frequencies must be determined, and genes should be grouped accordingly. For a total number of genes M, and a maximal number of classes k, the algorithm groups successive classes until the total frequency of each group exceeds M/k, the last class clumped to the next-to-last, if its frequency is smaller than $M/(2k)$. The algorithm usually outputs less than k classes that are reasonably well balanced. The equivalent frequency of a class is the mean over the class of all frequencies from the empirical distribution. Once the binning has been made, the inclusion probability of each gene is defined to be proportional to the equivalent frequency of its class, then normalized so that all probabilities add up to 1. The choice of k will be discussed in subsection 4.4.2.

It should be noted that the gene population consists of the N genes present in the vector of numeric values. The gene sets in the database are necessarily reduced to those N genes. Let M be the number of common gene symbols between the vector and the database. The grouping takes place in the "reduced" database. If $N > M$, that is there are genes in the vector that are not present in the database then these $N - M$ genes form a supplementary group with inclusion probability 0. We have also made the simplification that the random variables $X_1, X_2, \ldots, X_n$ are i.i.d. Although we will not prove it rigorously, we believe that this simplification (as if the genes were sampled with replacement) is valid since $N \gg n$. Finally, we are aware that the null hypothesis depends on the data and therefore the theoretical result does not strictly apply.

3.3 Monte-Carlo validation

In this section, a validation of the theoretical results derived above is provided. Firstly, the minimum gene set size for which the (D)WKS test can be applied with good precision is determined. Next, the output of Algorithm 1 is evaluated in cases where the limiting distribution can be calculated explicitly.

3.3.1 Validation of asymptotics on simulated data

Since the WKS test relies on a convergence theorem, it is necessary to determine the values of n (the gene set size) for which the procedure yields precise enough results. Such a validation is standard. For a given n, a sample of gene sets of size n is simulated, under the null hypothesis. For each of them, the test statistic is computed, thus a sample of values of the test statistic under the null hypothesis is obtained. The goodness-of-fit of the theoretical CDF F to the empirical CDF of the sample is tested by the (classical) KS test. Figure 3.5 shows results that were obtained for two functions g: one is $g_1(x) = 2(1-x)$ (left panel), the other one comes from real data: a sample in GSE36133 of [10] (right panel). The evaluation of F_1 was done over 10^6 Monte-Carlo simulations, and 10^5 discretization points, as explained in subsection 3.2.1. For the real data, the number of discretization points was $m = N = 18\,638$, and the number of Monte-Carlo simulation was $nsim = 20\,000$. The values of n range from 5 to 1 100 by step 5. For each n, 1 500 uniform random gene sets of size n were simulated. The negative logarithm in base 10 of the KS p-value is plotted. On each plot the horizontal line corresponding to a 5% p-value has been added. The p-values are small until $n = 40$, they stay above 5% after. This is coherent with what is observed for most asymptotic tests, and in particular the classical KS test.

On Figure 3.5, there is no clear difference between the theoretical g (left), and real data (right). However, it must be recalled that the null hypothesis $\mathcal{H}_0$, under which simulations have been done in

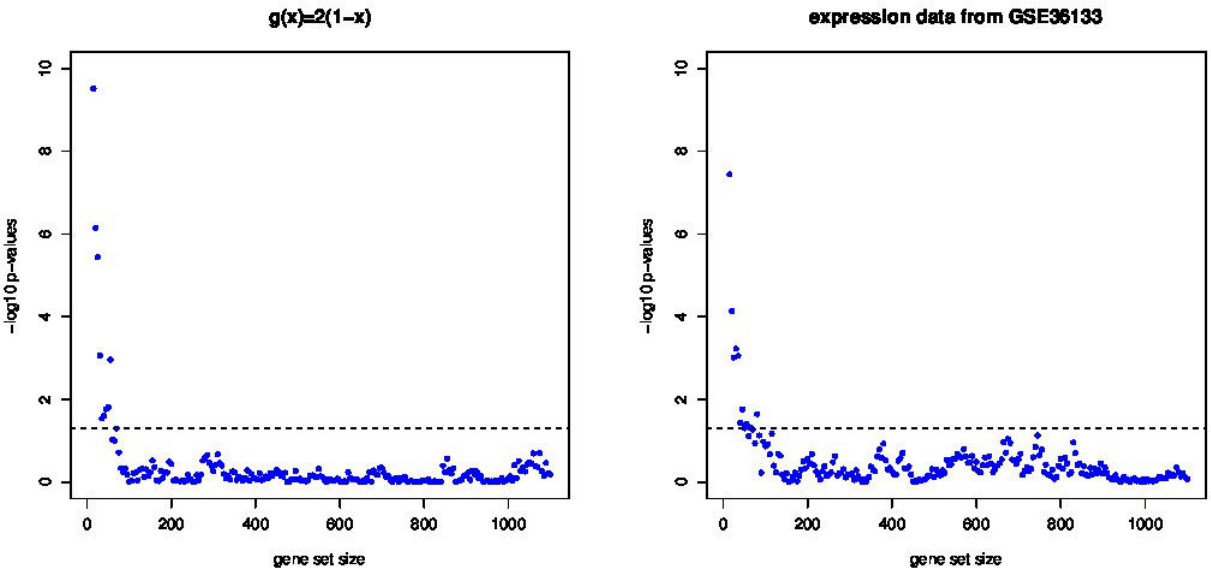


Figure 3.5: Goodness-of-fit of simulated WKS test statistic T_n^* over simulated gene sets. The function g is $g(x) = 2(1-x)$ on the left panel. It comes from real data on the right panel. The gene set size (abscissa) ranges from 5 to 1 100 by step 5. For each n the ordinate is the negative logarithm in base 10 of the KS goodness-of-fit p-value, over a sample of 1 500 gene sets. The dashed lines have ordinate $-\log_{10}(0.05)$.

both cases, is that the gene set is a sample of uniform random variables on the interval $[0, 1]$. However, in practice, the gene set should be considered instead as a random subset without replacement of the set of all genes. If the gene set size n is small compared to the total number of genes N, the difference is negligible. We have conducted another set of experiments, where gene sets were simulated by extracting random samples without replacement from $\{1/N, \ldots, N/N\}$. The results are depicted on Figure 3.6 and show a good agreement with those of Figure 3.5, until $n = 1\,000$. Beyond that value, the asymptotics becomes less precise. It must be observed that gene sets of size larger than 1 000 are relatively rare (28 out of the 4 722 gene sets of C2) in curated gene set databases but frequent in experimentally derived gene sets.

Beyond statistical validation, the comparison of the exact CDF, estimated over random gene sets, with the theoretical asymptotic F reveals an interesting feature of the WKS test: the exact CDF tends to be smaller than F. This implies that the asymptotic p-value tends to be larger than the true one, or else that the procedure is conservative: small gene sets are less likely to be declared significant by WKS (see Figure 3.7). However, if a better precision is needed for gene set sizes smaller than 40, then an estimation of the distribution of T_n^* through Monte-Carlo simulation should be preferred.

Finally, in order to validate the asymptotics in the general case (DWKS), an experiment similar to the ones described before was conducted. Figure 3.8 shows the results that were obtained for a function g coming from real data: a sample in GSE36133 and theoretical distribution function F, the beta distribution with parameters $\alpha = \frac{1}{2}$ and $\beta = \frac{1}{2}$. For the evaluation of the asymptotic distribution, the number of discretization points was $m = N = 18\,638$, and the number of Monte-Carlo simulations was $nsim = 10\,000$. The values of n range from 5 to 1 000 by step 5. For each n, 2 000 random gene sets from the beta distribution Beta$(\frac{1}{2}, \frac{1}{2})$ of size n were simulated. The negative logarithm in base 10 of the KS p-value is plotted. As before, the p-values are small until $n = 40$, they stay above 5% after.

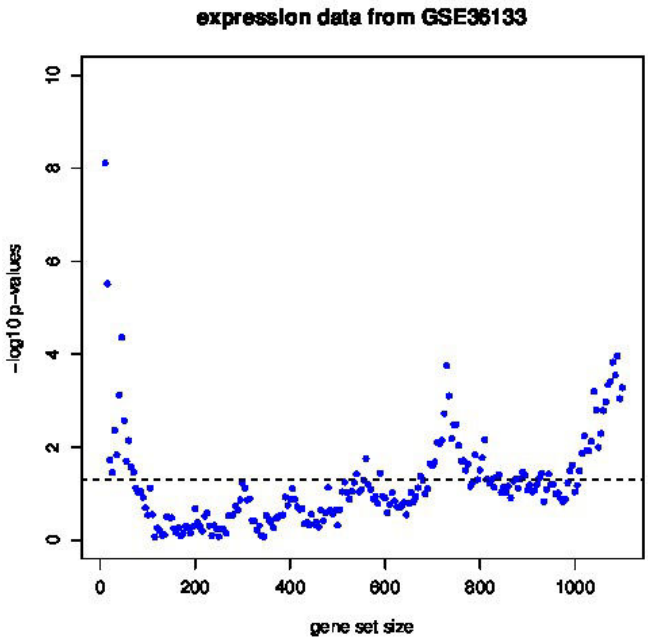


Figure 3.6: Goodness-of-fit of simulated WKS test statistic T_n^* over simulated gene sets, extracted as random samples without replacement from $\{1/N, \ldots, N/N\}$. The function g comes from real data. The gene set size (abscissa) ranges from 5 to 1 100 by step 5. For each n the ordinate is the negative logarithm in base 10 of the KS goodness-of-fit p-value, over a sample of 1 500 gene sets. The dashed lines have ordinate $-\log_{10}(0.05)$.

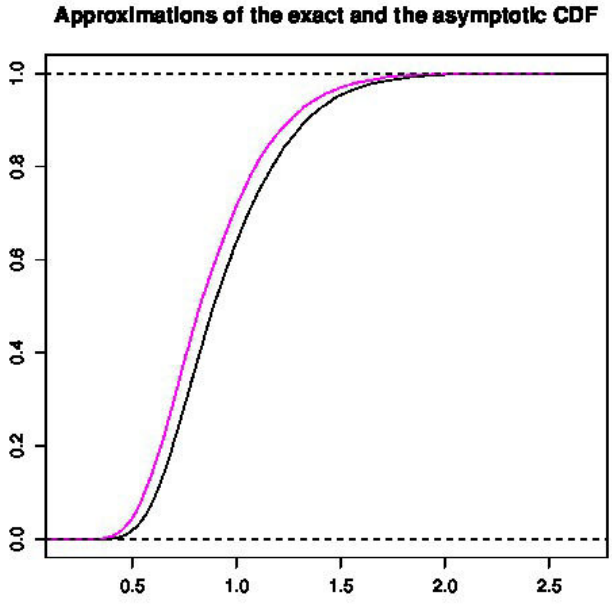


Figure 3.7: Exact CDF of T_7^*, estimated over random gene sets (pink curve), together with the theoretical asymptotic F (black curve).

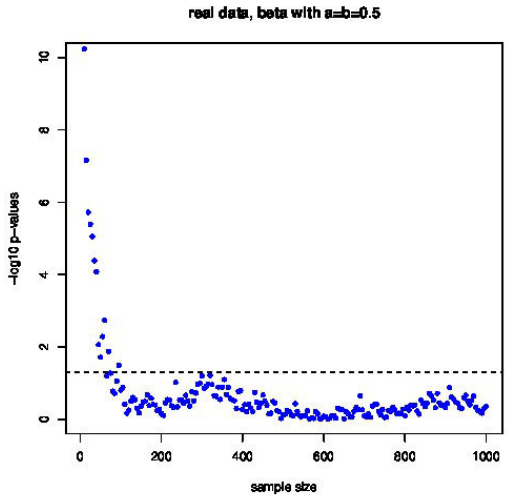


Figure 3.8: Goodness-of-fit of simulated DWKS test statistic over simulated gene sets. The function g comes from real data. The gene set size (abscissa) ranges from 5 to 1 000 by step 5. For each n the ordinate is the negative logarithm in base 10 of the KS goodness-of-fit p-value, over a sample of 2 000 gene sets. The dashed lines have ordinate $-\log_{10}(0.05)$.

3.3.2 Checking theoretical results on simulation

Since many approximations have been used in Algorithm 1 for the evaluation of the cumulative distribution function F defined by (3.1.13), its outcome must be evaluated. We validate its output in two cases where an explicit calculation for the limiting distribution is available: $g(\cdot) = 1$ and $g(\cdot) = 2\mathbb{I}_{[0,\frac{1}{2}]}(\cdot)$. The results are shown on Figure 3.9. In each plot, the exact CDF (blue curve) and the estimated CDF (black curve) are superposed. The black curves were obtained through 10 000 Monte-Carlo simulations. The two curves are indistinguishable. To be more precise, the maximal absolute difference between the exact curve and its estimation is 0.0105 (left figure) and 0.0111 (right figure). If we apply the one-sample KS test, we obtain $p = 0.218$ in the first case and $p = 0.168$ in the second.

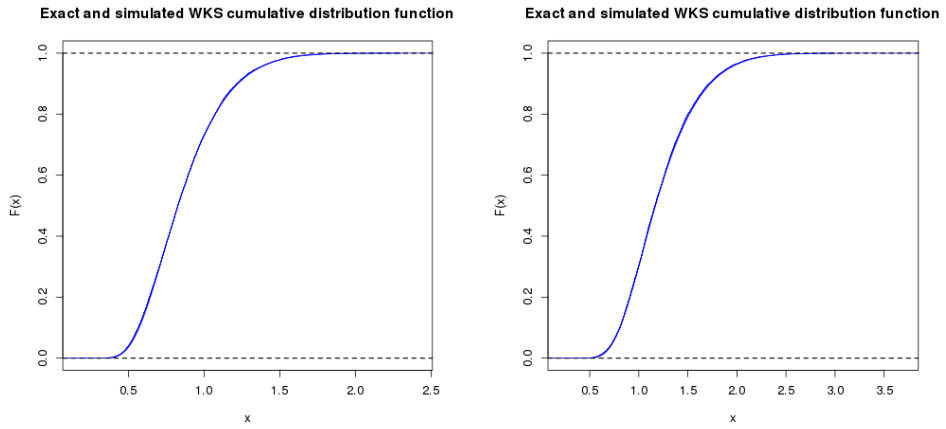


Figure 3.9: KS distribution (blue curve) and estimated CDF (black curve) from application of Algorithm 1. Exact WKS CDF corresponding to $g(x) = 2\mathbb{I}_{[0,\frac{1}{2}]}(x)$ and estimated CDF (black curve) from application of Algorithm 1.

Chapter 4

Applications

4.1 Statistics on databases

The Molecular Signatures Database with the 7 major collections and subcollections, that was presented in Chapter 2, will be systematically used in the experiments to follow. For this reason, first of all, Table 4.1 summarizes some basic statistics on these lists of gene sets: number of gene sets (NbGS), median gene set size (MS), number of different genes (NG), and median gene frequency (MGF).

Database	Type of Data	NbGS	MS	NG	MGF
C1	positional gene sets	326	65.5	30 010	1
C2	curated gene sets	4722	39	21 047	13
C3	motif gene sets	836	233.5	14085	8
C4	computational gene sets	858	62	10 061	6
C5	GO gene sets	1454	27.5	8 278	12
C6	oncogenic signatures	189	180	11 250	2
C7	immunologic signatures	1910	200	19 841	15
Biocarta	BioCarta gene sets	217	18	1 267	1
Kegg	KEGG gene sets	186	53	5 267	1
GOBP	GO biological process	825	28	6 178	9
GOCC	GO cellular component	233	28	5 270	5
GOMF	GO molecular function	396	27	5 314	3

Table 4.1: Some genomics databases, with number of gene sets (NbGS), median gene set size (MS), number of different genes (NG), and median gene frequency (MGF).

Figure 4.1 (left panel) represents the gene frequency distribution in the C2 database: the distribution is very much skewed to the right, with a large number of genes represented only once or twice, while some are present in more than 100 gene sets. This is typical of what is observed in all databases (see Figure 4.2) but C1. C1 is a particular database and will not be considered in what follows: it looks like a geographical map of the genes with almost every gene being present in only one gene set. In order to test the appropriateness of the null hypothesis that the gene sets are random samples without replacement from the total gene population (here all genes in C2), we simulated a database with the same size and gene set sizes as those of C2 but with randomly sampled gene sets. The gene frequency distribution in the simulated database is shown on Figure 4.1 (right panel): the distribution is close to Gaussian, with few different frequencies. In Figure 4.3, we have plotted the gene frequency distribution in 3 databases derived from C2 by keeping:

- the gene sets related to cancer
- the gene sets with the word “breast” in their name

- the gene sets with size > 200

The same observations as before can be made.

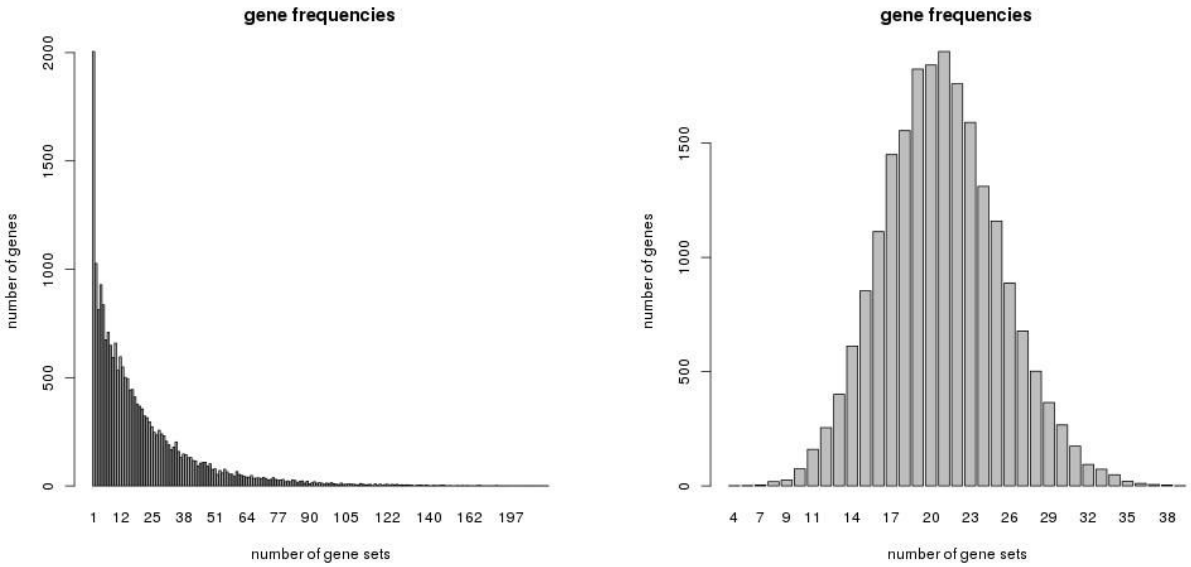


Figure 4.1: Gene frequency distribution in the C2 database (left) and in a simulated database with the same size and same gene set sizes, each gene set being a random sample without replacement from the total gene population (right).

From Figure 4.1, it is obvious that the two distributions are very different from each other. Next, we repeated the procedure in order to test the appropriateness of the null hypothesis:

$\mathcal{H}_0$*: The gene sets are samples without replacement from the gene population with the genes being included with different probabilities*

which is the basis for the model proposed in Chapter 3 (subsections 3.1.3 and 3.2.4). More specifically, the genes in the C2 database were divided into two groups and an inclusion probability for each one was estimated by application of the algorithm described in subsection 3.2.4. Then, a database was simulated with the same size and same gene set sizes as those of C2 but where the gene sets are samples without replacement from the gene population with the genes being included with probability equal to their inclusion probability. To be more precise, the samples were drawn according to the Wallenius model [46]: the probability of taking a particular gene at a particular draw is equal to this gene's fraction of the total probability of all genes that have not been drawn so far. The gene frequency distribution was then defined. The same procedure was repeated for 6 groups of genes. Finally, the procedure was repeated for samples where the genes were included with probability proportional to their frequency in the database (Wallenius model). The results are depicted on Figure 4.4. Although there might still be discrepancies, the distribution in Figure 4.4 (right panel) is much closer to what is actually observed (see left panel of Figure 4.1).

In order to assess more formally the discrepancies observed above, the following theoretical approach was adopted: Let us first consider the model of equal probability sampling without replacement and assume that the gene sets are independent samples without replacement of the total population of N genes. Then, the probability of a sample of size n to contain a given gene is:

$$p_n = \frac{n}{N} \ .$$

We are interested in the number of gene sets containing a given gene. Let Z be that number. Let k be the number of gene sets. For $i = 1, \ldots, k$ let n_i denote the size of the i-th gene set. Let B_i denote

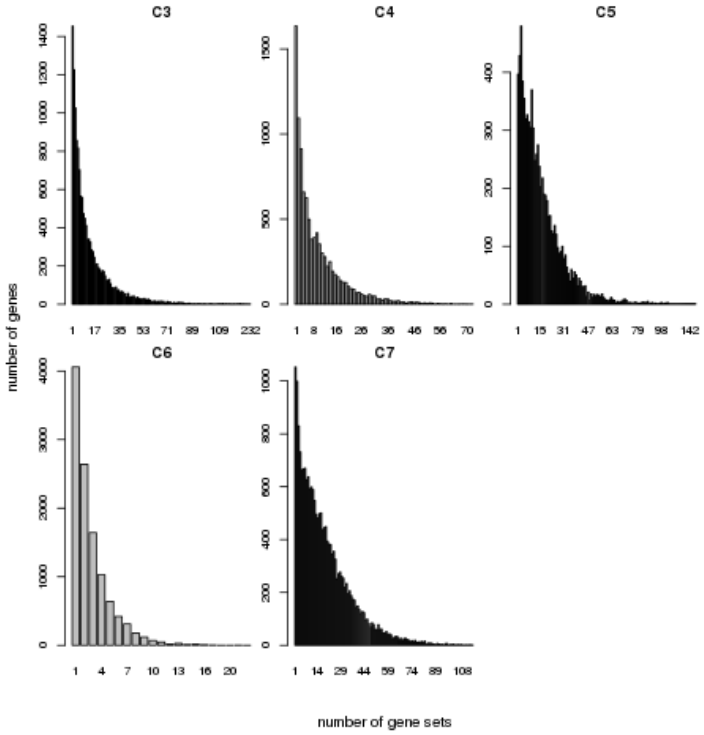


Figure 4.2: Gene frequency distribution in databases C3, C4, C5, C6 and C7.

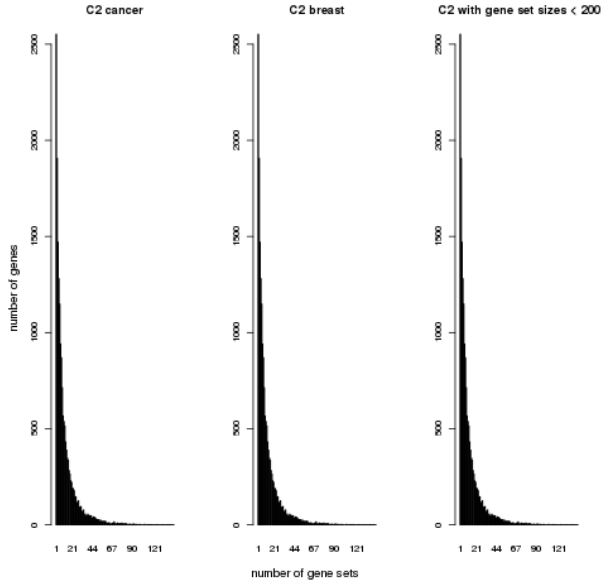


Figure 4.3: Gene frequency distribution in some "reduced" C2 databases: gene sets related to cancer (left panel), gene sets related to breast cancer (middle panel) and gene sets with size < 200 (right panel).

the Bernoulli random variable taking the value 1 or 0 according to whether the given gene belongs to the i-th gene set or not. Then Z is the sum of the binary random variables B_i's. If all gene sets are

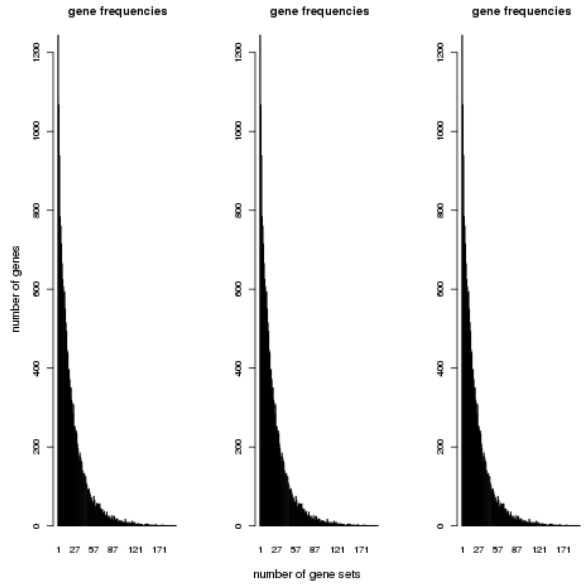


Figure 4.4: Gene frequency distribution in a simulated database with the same size and same gene set sizes as those of C2 but with gene sets that are samples without replacement, where the genes are included with probability proportional to a given inclusion probability. We have two groups (left panel), 6 groups (middle panel). In the right panel, the genes genes are included with probability proportional to their frequency in the database.

independent equal probability random samples without replacement, then the B_i's are i.i.d. random variables, with B_i having Bernoulli distribution with parameter p_{n_i}. The distribution of a sum of independent Bernoulli random variables with different parameters is called "Poisson-binomial":

$$\text{Prob}(z) = \prod_{i=1}^{k} (1 - p_{n_i}) \sum_{i_1 < \cdots < i_z} w_{i_1} \cdots w_{i_z} \ , \quad z \in \{0, \ldots, k\} \ , \tag{4.1.1}$$

where $w_i = \frac{p_{n_i}}{1-p_{n_i}}$ for $i = 1, \ldots, k$ and the summation is over all possible combinations of distinct $i_1, \ldots, i_z$ from $\{1, \ldots, k\}$. If k is large and the p_{n_i}'s are relatively small (which is the case here), it can be approximated by a Poisson distribution with parameter

$$\lambda = \sum_{i=1}^{k} p_{n_i} \ .$$

A generalization of the above analysis for a group of k given genes ($k \geqslant 1$) can be found in [126]. Now, we are going to repeat the same procedure and give an analytical result for the distribution of Z, when we have two groups of genes. Denote by N_j ($j = 1, 2$) the number of genes of group j and by π_j its inclusion probability. Then, regarding the distribution of Z we can write:

$$\begin{aligned} \mathbb{P}[Z = z] &= \mathbb{P}[Z = z | \text{gene} \in \text{group 1}] \mathbb{P}[\text{gene} \in \text{group 1}] \\ &+ \mathbb{P}[Z = z | \text{gene} \in \text{group 2}] \mathbb{P}[\text{gene} \in \text{group 2}] \\ &= \mathbb{P}[Z = z | \text{gene} \in \text{group 1}] \frac{N_1}{N} + \mathbb{P}[Z = z | \text{gene} \in \text{group 2}] \frac{N_2}{N} \ . \end{aligned}$$

But, if we assume for simplicity that the gene has the same probability to appear in all positions of the gene set, then the conditional distribution $\mathbb{P}[Z = z | \text{gene} \in \text{group 1}]$ is nothing else but a

Poisson-binomial distribution that can be approximated by a Poisson distribution with parameter

$$\lambda = \sum_{i=1}^{k} n_i \pi_1$$

and the conditional distribution $\mathbb{P}[Z = z|\text{gene} \in \text{group 2}]$ is a Poisson-binomial distribution that can be approximated by a Poisson distribution with parameter

$$\lambda = \sum_{i=1}^{k} n_i \pi_2 \ .$$

This result can be easily generalized to more than two groups. We have repeated the simulation of Figure 4.1 (right panel) and that of Figure 4.4 (left panel) and superposed the theoretical curves, calculated from the above relations. The result is shown on Figure 4.5.

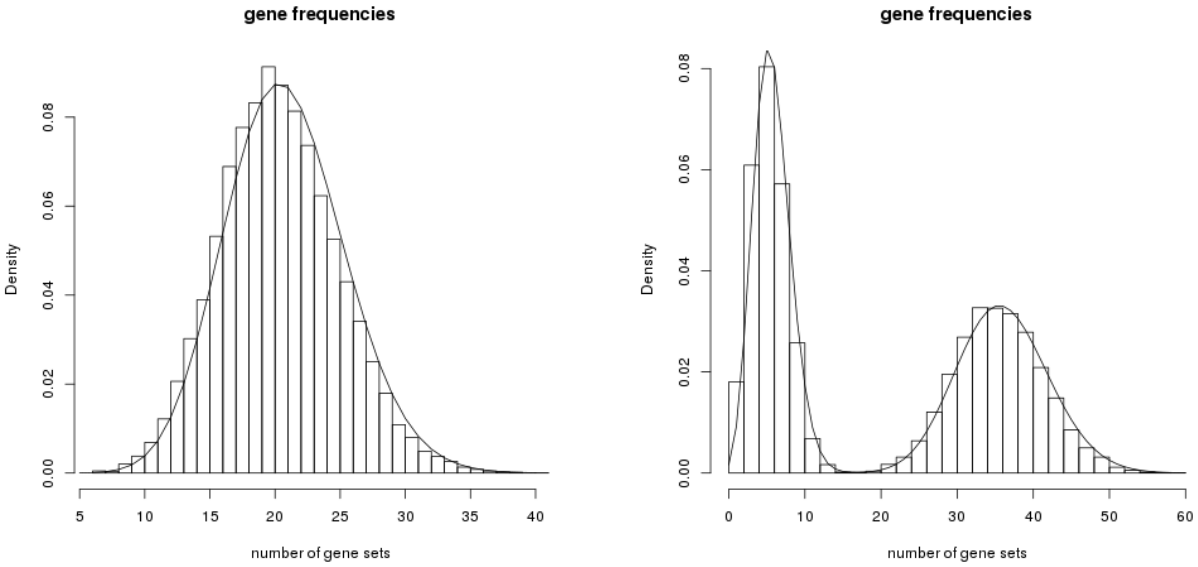


Figure 4.5: Gene frequency distribution in a simulated database with the same size and same gene set sizes as those of C2, each gene set being a random sample without replacement from the total gene population with the theoretical distribution superposed (left panel). Gene frequency distribution in a simulated database with the same size and same gene set sizes as those of C2, each gene set being a sample without replacement, where the genes are included with probability proportional to a given inclusion probability (two groups of genes) with the theoretical distribution superposed (right panel).

Then, the question if the "frequent" genes were common to all databases had to be answered. First of all, the gene frequencies were calculated for the databases C2, C3, C4, C5, C6, C7. This yielded 6 vectors of the appropriate size. All pairwise correlations between the vectors were computed, after reducing every two vectors to the common gene symbols: $r_{\text{C2,C3}} = 0.199$, $r_{\text{C2,C4}} = 0.276$, $r_{\text{C2,C5}} = 0.136$, $r_{\text{C2,C6}} = 0.358$, $r_{\text{C2,C7}} = 0.604$, $r_{\text{C3,C4}} = 0.028$, $r_{\text{C3,C5}} = 0.041$, $r_{\text{C3,C6}} = 0.076$, $r_{\text{C3,C7}} = 0.089$, $r_{\text{C4,C5}} = 0.067$, $r_{\text{C4,C6}} = 0.093$, $r_{\text{C4,C7}} = 0.136$, $r_{\text{C5,C6}} = 0.004$, $r_{\text{C5,C7}} = 0.042$ and $r_{\text{C6,C7}} = 0.217$. All these correlations can be regarded as significant at significance level 0.05, apart from $r_{\text{C5,C6}}$. In order to have a graphical representation, scatterplots of gene frequencies in one database vs another for all different combinations of databases have been plotted. The results are shown on Figure 4.6.

From Figure 4.6, we see that there is usually a positive correlation between the gene frequencies in one database and those in another: genes that are frequent in one database tend to be frequent

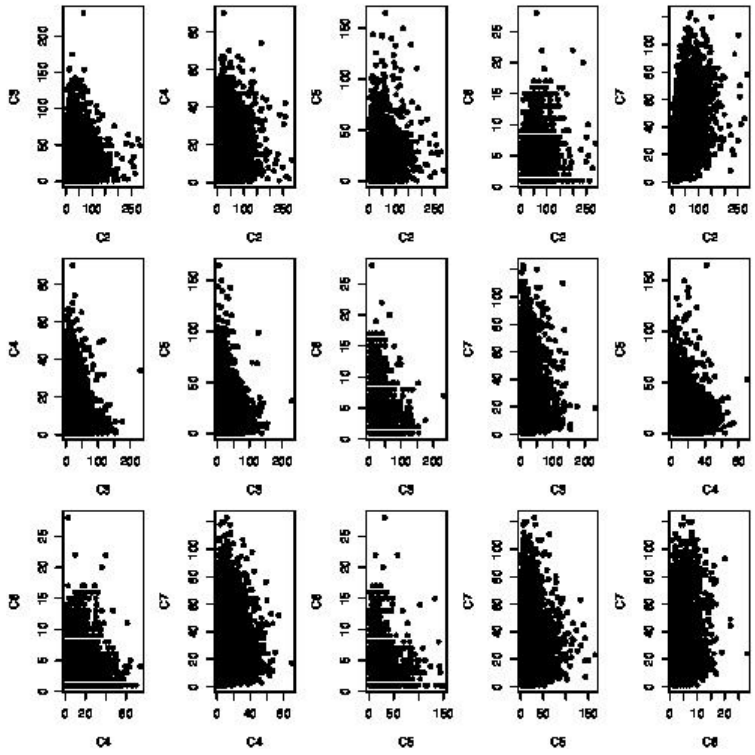


Figure 4.6: Scatterplot of gene frequencies in one database vs another for all different combinations of databases.

in another and similarly for the unfrequent. However, the correlation coefficients vary. There is a strong relation between the databases C2 and C7, while there is a very weak (no) relation between the databases C5 and C6.

Next, the 6 vectors containing the gene frequencies in the databases C2, C3, C4, C5, C6 and C7 were reduced to the common gene symbols. A principal component analysis (PCA) was computed by considering the genes as individuals and the databases as variables. The result is shown on Figure 4.7. The pca structure is given on Table 4.1.

	PC1	PC2	PC3	PC4	PC5	PC6
C2	-0.91	0.04	0.21	-0.34	0.10	-0.03
C3	-0.12	-0.97	-0.19	0.09	-0.03	0.00
C4	-0.08	0.02	0.07	-0.05	-0.99	0.00
C5	-0.08	-0.08	0.737	0.67	0.02	0.01
C6	-0.02	0.00	0.00	-0.012	0.00	0.99
C7	-0.38	0.23	-0.61	0.65	-0.04	0.00

Table 4.2: Pca structure from the analysis of the gene frequencies in 6 databases: C2, C3, C4, C5, C6, C7.

From Table 4.1, we see that there is a very strong correlation between the database C2 and the first principal component. The first principal component is also correlated with the database C7. This means that genes with high values will be frequent both in the databases C2 and C7. The second principal component is strongly correlated with the database C3 only. This means that genes with high values will have high frequency only for the database C3. On Figure 4.7, we have identified 3 gene symbols: JUN, CDKN1A and DMD. The first two are very frequent both in the databases C2

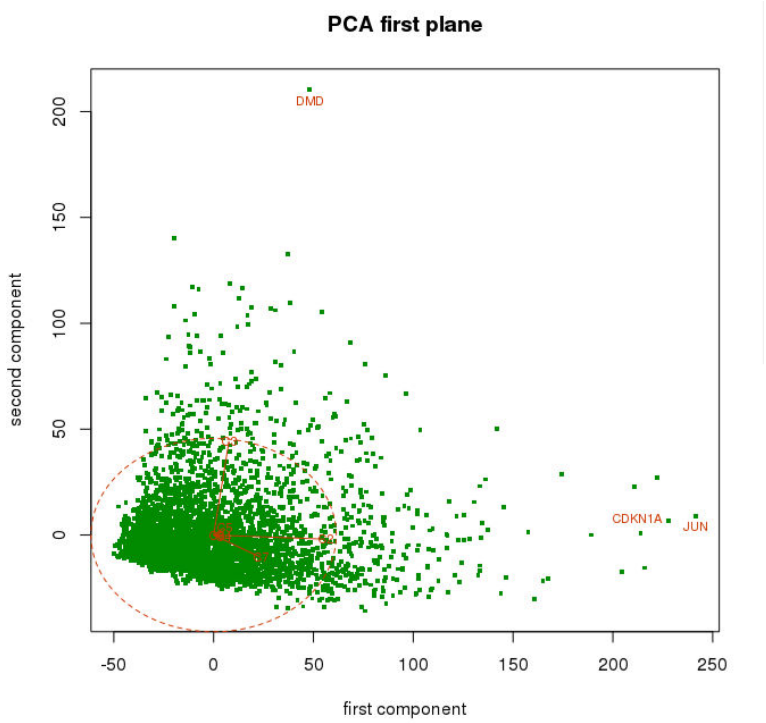


Figure 4.7: Principal component analysis on the gene frequencies in 6 databases: C2, C3, C4, C5, C6, C7. Samples (genes) are represented as points in green.

and C7, while the last one is frequent only in C3.

4.2 Statistics on expression matrices

In this section, the relation between the median (normalized) ranks of genes in expression matrices and their frequency in databases is studied.

Using the R script performing the treatments described in subsection 2.2.1 (see subsection 4.6.2 and Appendix), we were able to download 17 datasets from the GEO repository: they are detailed in Table 4.2. They were selected on a criterion of size (number of samples 500 or more). The 17 matrices together amount to 12 251 samples. To each study, a three-letter acronym was attached.

Merging different datasets, requires prior checking that the information they contain is compatible. An obvious obstacle to simultaneous treatment is that expression data collected under different experimental conditions and/or platforms usually have incompatible distributions, which differ sometimes by several orders of magnitude [66, 72, 83]. A solution is provided by robust (or distribution-free) statistics [51, 54]. Robust methods amount to replacing actual values by ranks, or equivalently by empirical distribution functions or van der Waerden's normal scores [51, p. 309]. This idea has already been applied to expression data in several papers, including [116, 118, 109, 19].

For this reason, at first each data matrix was transformed by replacing its column values by the empirical distribution function. Then, for each dataset, the median of all rows was computed. This yielded 17 vectors of size 15 190, the correlation matrix of which is given in Table 4.2. A positive (negative) correlation between two vectors of size 15 190 is significant at threshold 5% if it is larger than 0.013 (smaller than -0.013); thus, all correlations of Table 4.2 can be regarded as significant.

From Table 4.2, we deduce that all the studies contain information that is compatible except for two: PVA which has negative correlations and PRS which has weak correlations. For this reason, these two studies will be exempted from the further analysis to be carried out. The above analysis

acronym	reference	GSE number	GPL number	symbols (rows)	samples (columns)
PMM	[27]	2658	570	20 184	559
AML	[34]	6891	570	20 184	537
HBI	[96]	7307	570	20 184	677
MDS	[88]	15061	570	20 184	870
MMD	[93]	24080	570	20 184	559
DLB	[47]	31312	570	20 184	498
PRS	[119]	33828	10558	20 768	881
CCL	[10]	36133	15308	18 722	917
BEC	[55]	36192	6947	19 628	911
WBS	[86]	36382	6947	19 628	991
GSC	[124]	36809	570	18 260	812
MBI	[100]	37069	570	18 260	590
CCC	[85]	39582	570	20 184	566
PVA	[120]	48152	6947	19 628	705
HPS	[119]	48348	6947	19 628	734
XMD	[57]	48433	570	20 184	823
HAV	[91]	48762	6947	19 628	621

Table 4.3: Seventeen GEO series have been chosen, coming from four different platforms. To each of them a three letters acronym was associated. The table gives the acronym, the GEO reference (GSE number), the reference of the platform (GPL number). For the data matrix, the number of symbols after annotation and reduction, and the number of columns (also called samples or GSM's) are given. All 17 data matrices had 15 190 gene symbols in common.

can be found in much greater detail in [125].

As explained at the beginning of the section, our aim is to see whether there is a relation between the median (normalized) rank in expression matrices and the database frequency or if these two random variables are independent. To answer this question, we merged the 15 "compatible" datasets and computed the median in each row of the resulting matrix. This yielded a vector of size 15 673 (common gene symbols without the studies PVA and PRS), which contains the median (normalized) ranks of the genes. Then, we also merged the databases C2, C3, C4, C5, C6, C7, obtaining one big database with 9 969 gene sets. The gene frequencies in the resulting database were also computed. This yielded a vector of size 22 656. The rank correlation between these two vectors, after reducing to common gene symbols (15 329), was calculated. It was 0.40, yielding a p-value equal to zero. In Figure 4.8, a scatterplot of the two vectors, after replacing the gene frequencies by the empirical distribution function is shown.

From Figure 4.8, we see that there is a positive correlation between median ranks in expression matrices and gene frequencies in the databases. More specifically, the "frequent" genes tend to have higher ranks and thus high expression levels, while the "not-frequent" have smaller expression levels. In Figure 4.9, the distribution of median normalized ranks for the 344 (15 673-15 329) genes present only in the expression matrices and not appearing in the "big" database (and thus in any of the individual databases) has been plotted. We see that they tend to have low expression levels. We would also like to note the fact that out of the 1 484 genes with frequency equal to 1 in the "big" database, only 45 appear in the merged expression matrices.

From the above analysis, we conclude that these two random variables (rank in the vector of numeric data and frequency in the database) are not independent but have a positive association. Thus, the issue of unequal gene frequencies must be taken into account.

	PMM	AML	HBI	MDS	MMD	DLB	PRS	CCL
PMM	1.00	0.75	0.79	0.81	0.63	0.55	0.19	0.74
AML	0.75	1.00	0.68	0.85	0.58	0.59	0.18	0.69
HBI	0.79	0.68	1.00	0.76	0.46	0.53	0.18	0.75
MDS	0.81	0.85	0.76	1.00	0.52	0.60	0.18	0.77
MMD	0.63	0.58	0.46	0.52	1.00	0.50	0.11	0.54
DLB	0.55	0.59	0.53	0.60	0.50	1.00	0.12	0.56
PRS	0.19	0.18	0.18	0.18	0.11	0.12	1.00	0.20
CCL	0.74	0.69	0.75	0.77	0.54	0.56	0.20	1.00
BEC	0.49	0.43	0.65	0.48	0.32	0.35	0.21	0.56
WBS	0.58	0.62	0.54	0.70	0.40	0.46	0.23	0.56
GSC	0.61	0.69	0.57	0.74	0.46	0.49	0.15	0.56
MBI	0.63	0.70	0.58	0.75	0.48	0.50	0.15	0.59
CCC	0.62	0.64	0.67	0.67	0.49	0.53	0.15	0.74
PVA	−0.10	−0.13	−0.09	−0.16	−0.09	−0.09	0.16	−0.08
HPS	0.62	0.66	0.58	0.74	0.43	0.49	0.24	0.61
XMD	0.81	0.72	0.84	0.83	0.51	0.56	0.19	0.92
HAV	0.60	0.64	0.56	0.73	0.42	0.48	0.21	0.60

	BEC	WBS	GSC	MBI	CCC	PVA	HPS	XMD	HAV
PMM	0.49	0.58	0.61	0.63	0.62	−0.10	0.62	0.81	0.60
AML	0.43	0.62	0.69	0.70	0.64	−0.13	0.66	0.72	0.64
HBI	0.65	0.54	0.57	0.58	0.67	−0.09	0.58	0.84	0.56
MDS	0.48	0.70	0.74	0.75	0.67	−0.16	0.74	0.83	0.73
MMD	0.32	0.40	0.46	0.48	0.49	−0.09	0.43	0.51	0.42
DLB	0.35	0.46	0.49	0.50	0.53	−0.09	0.49	0.56	0.48
PRS	0.21	0.23	0.15	0.15	0.15	0.16	0.24	0.19	0.21
CCL	0.56	0.56	0.56	0.59	0.74	−0.08	0.61	0.92	0.60
BEC	1.00	0.64	0.37	0.37	0.45	−0.13	0.65	0.57	0.59
WBS	0.64	1.00	0.64	0.63	0.49	−0.29	0.95	0.57	0.88
GSC	0.37	0.64	1.00	0.94	0.61	−0.19	0.66	0.57	0.64
MBI	0.37	0.63	0.94	1.00	0.62	−0.17	0.66	0.59	0.64
CCC	0.45	0.49	0.61	0.62	1.00	−0.09	0.53	0.75	0.51
PVA	−0.13	−0.29	−0.19	−0.17	−0.09	1.00	−0.27	−0.08	−0.26
HPS	0.65	0.95	0.66	0.66	0.53	−0.27	1.00	0.62	0.90
XMD	0.57	0.57	0.57	0.59	0.75	−0.08	0.62	1.00	0.60
HAV	0.59	0.88	0.64	0.64	0.51	−0.26	0.90	0.60	1.00

Table 4.4: For each of the 17 data matrices of Table 4.2, after replacing its column values by the empirical distribution function, the median row value of each gene symbol was computed. This gave 17 vectors with length 15 190 (number of common symbols). The table gives pairwise correlations between the 17 vectors.

4.3 Comparison of statistical tests in terms of power

In this section, a comparison of statistical tests in terms of power will be carried out. The following tests have been considered: PAGE [70], WKS, WKSr (WKS where the values have been replaced by ranks) and KS (WKS for $g \equiv 1$, see subsection 3.1.1). For the method PAGE, three subcases were analyzed: firstly the gene set is identified by the sample $(U_1, \ldots, U_n)$, then by $(g(U_1), \ldots, g(U_n))$ and finally by $(g(U_1)U_1, \ldots, g(U_n)U_n)$ (same notations as in subsection 3.1.1). We will be referring to the first subcase as PAGEr, to the second as PAGEo and to the third as PAGEp. We chose to compare methods where (asymptotic) analytical results were available for the null distribution. For this reason, GSEA will not be taken into account. Other analytical methods like GAGE [81], the method by Boorsma et al. [16] or the method by Irizarry et al. [64] are very close to PAGE and thus will be exempted from the analysis too. Finally, the DWKS test will be treated afterwards separately, because it is based on the data.

Four parametered families of alternatives $\mathcal{H}_p$ were studied. In each family, power functions were computed in the following way: for each parameter p, 1 000 samples of size 100 and 200 were simulated according to $\mathcal{H}_p$ and the power for each test was estimated as the proportion of p-values below 5%. As a vector of numerical values, a sample of expression levels from the GEO dataset GSE36133 [10] was chosen. The parametered families were the following:

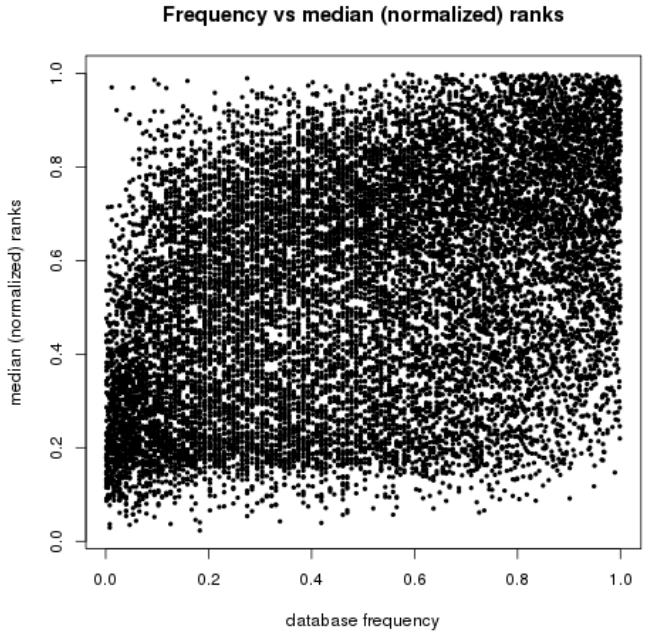


Figure 4.8: Scatterplot of gene frequencies vs median (normalized) ranks. The gene frequencies have been replaced by the empirical distribution function. Each point corresponds to a gene, the abscissa being its frequency in the merged database and the ordinate being its median normalized rank in the expression matrices.

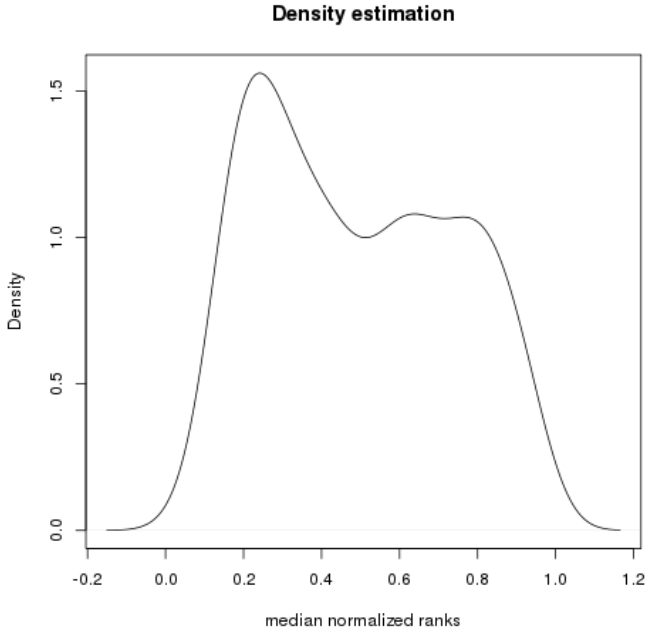


Figure 4.9: Distribution of median normalized ranks for the 344 genes present only in the expression matrices and not appearing in the merged database.

1. $\mathcal{H}_p$: *The genes are included in the gene set with probability proportional to* $(1-p)+pg$

where $p = \frac{1}{100}, \frac{2}{100}, \ldots, 1$. The power functions so obtained are depicted on Figure 4.10. From Figure 4.10, it follows that PAGE does better than the WKS or WKSr test in terms of power. However, this is not necessarily bad for the WKS test. As it has already been discussed, in real applications many gene sets contain the "frequent" genes that tend to appear in the first part of the ranked vector and thus end up being declared as significant but are actually false discoveries. The hypotheses of this family try to imitate the real application: genes that have been ranked first in the vector have a higher probability to be included in the gene set. Although the WKS test was not constructed to address this problem, we see that it does partially so. A similar family of hypotheses, where $g(x) = 5(1-x)^{\frac{1}{4}}$ was also studied. This function was chosen for two reasons: firstly, because it is an analytical function (not based on the data) and secondly because as shown in subsection 3.2.1, it is a very steep function that will assign a much bigger probability to the first gene of the ranked vector compared to that of the last gene and we wanted to check if this would change the result.

$\mathcal{H}_p$: *The genes are included in the gene set with probability proportional to* $(1-p)+p5(1-x)^{\frac{1}{4}}$

The results are presented on Figure 4.11. The same observations as above can be made.

2. $\mathcal{H}_p$: *The gene set is a random sample without replacement out of the first p genes of the ranked vector*

where p ranges from N to 1 000 in such a way that 100 equidistant points are obtained, and N denotes the length of the vector. The results are shown on Figure 4.12. In this example, we see that the KS test and PAGE perform very well, while WKS and WKSr perform a little bit worse. We will try to give an intuitive explanation why WKS behaves a little bit worse than KS. Since the primitive function G lies above the bisector, then the maximal difference between the cumulated weight function and the primitive function G will probably be smaller than the maximal difference between the ECDF and the bisector (consider the case where the gene set contains the first 100 genes of the ranked vector). In addition, as it was shown in subsection 3.2.1, the WKS distributions have heavier tails than the KS distribution.

3. $\mathcal{H}_p$: *Half of the genes of the gene set are randomly sampled without replacement out of the first p genes of the ranked vector and the rest out of the last p genes.*

where p ranges from $\frac{N}{2}$ to 1 000 in such a way that 100 equidistant points are obtained. The power functions have been plotted on Figure 4.13. This family was chosen to show one major limitation of PAGE (and especially of PAGEr). As expected, PAGEr cannot detect any such gene set. On the contrary, WKSr and WKS outperform all the other statistical tests. This is very important, since it shows that WKS test can detect bi-directional changes when the weights are positive. WKS succeeds to do so in the following way: Half of the genes are agglomerated at the top of the ranking and half at the bottom. The function g gives much more weight to the first clustering than the second and as a result the maximal difference between the cumulated weight function and the primitive function G is very big.

4. In this case, the expression levels were replaced by the absolute values of a random sample of size N from normal distribution. In this way, we could consider that the weights represent "fold" changes (FC). Next, the vector was ranked in decreasing order of values and the genes were divided into two categories: those with FC $\geqslant 1.5$ that will be called "interesting" genes and those with FC < 1.5 that will be the "not interesting" genes. Let $p_0 = \frac{\#\text{"interesting" genes}}{\#\text{"not interesting" genes}}$. The family considered was the following:

$\mathcal{H}_p$: *The gene set contains a proportion of p "interesting" genes and* $1-p$ *"not interesting" genes.*

where p ranges from p_0 to 1 in such a way that 100 equidistant points are obtained. The results are displayed on Figure 4.14. From Figure 4.14, it is obvious that all 5 tests perform very well, with the WKS and WKSr behaving a little bit better. This means that if the proportion of "interesting" genes over the "not interesting" genes inside a gene set is slightly bigger than the proportion of the total number of "interesting" genes over the total number of "not interesting" genes, then the WKS test has a bigger chance to detect the corresponding gene set.

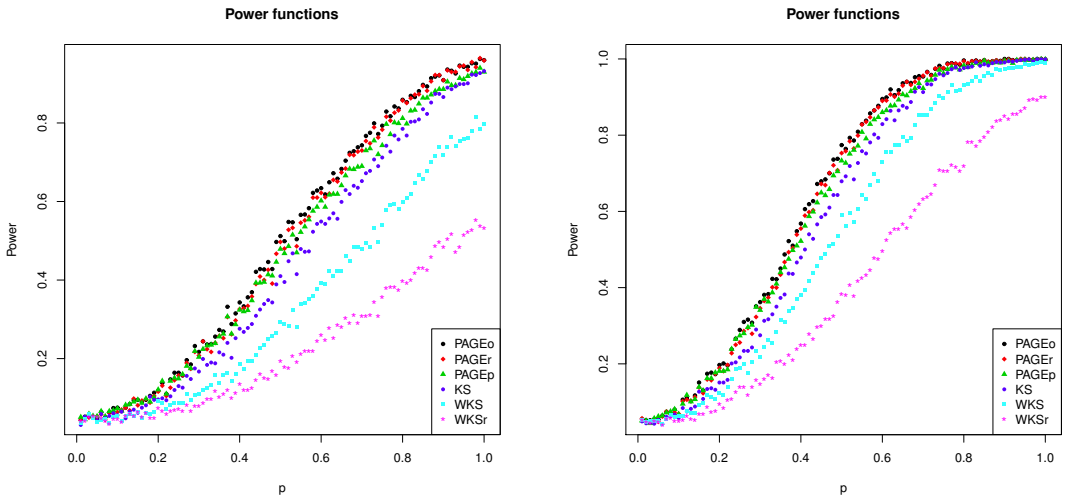


Figure 4.10: Power functions of the tests PAGEo, PAGEr, PAGEp, KS, WKS and WKSr for the first parametered family and sample size $n = 100$ (left panel). Power functions of the same tests for the first parametered family and sample size $n = 200$ (right panel). In each plot, the parameter p (abscissa) ranges from $\frac{1}{100}$ to 1 by step $\frac{1}{100}$. Six points correspond to each p, the ordinates of which are the power of the six tests: the point is represented by a black circle in the case of PAGEo, a red rhombus for PAGEr, a green triangle for PAGEp, a blue circle for KS, a light blue square for WKS and a pink star for WKSr.

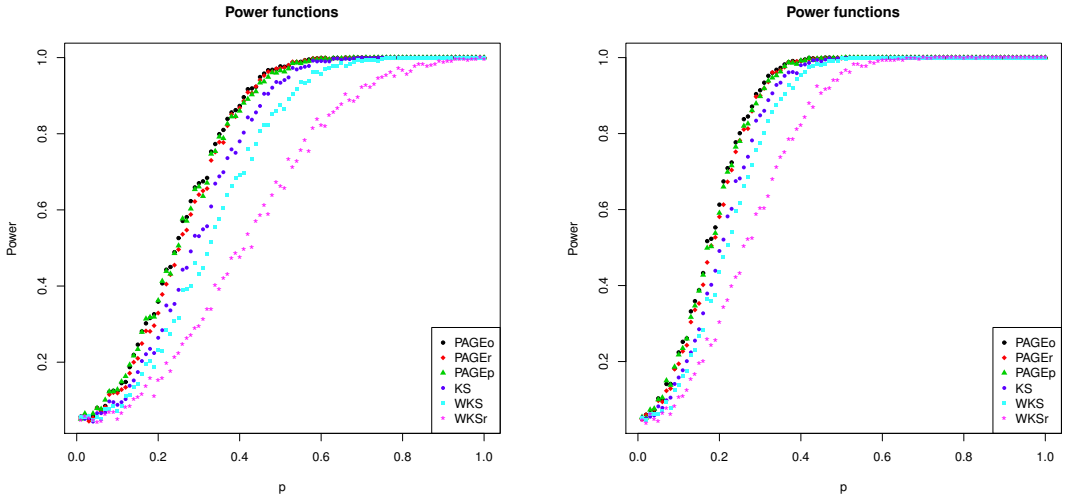


Figure 4.11: Power functions of the tests PAGEo, PAGEr, PAGEp, KS, WKS and WKSr for the first parametered family for an analytical function g ($g(x) = 5(1-x)^{\frac{1}{4}}$) and sample size $n = 100$ (left panel). Power functions of the same tests for the first parametered family for an analytical function g and sample size $n = 200$ (right panel). In each plot, the parameter p (abscissa) ranges from $\frac{1}{100}$ to 1 by step $\frac{1}{100}$. Six points correspond to each p, the ordinates of which are the power of the six tests: the point is represented by a black circle in the case of PAGEo, a red rhombus for PAGEr, a green triangle for PAGEp, a blue circle for KS, a light blue square for WKS and a pink star for WKSr.

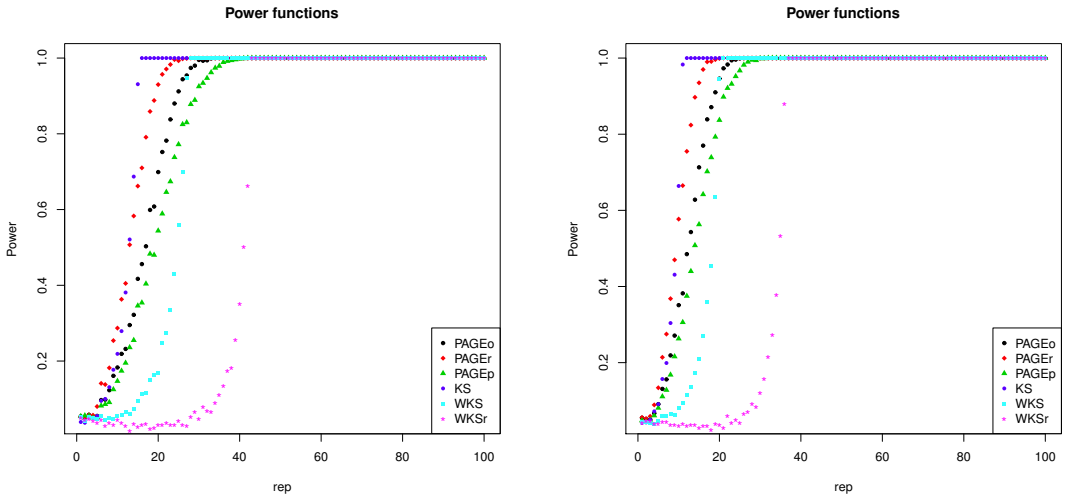


Figure 4.12: Power functions of the tests PAGEo, PAGEr, PAGEp, KS, WKS and WKSr for the second parametered family and sample size $n = 100$ (left panel). Power functions of the same tests for the second parametered family and sample size $n = 200$ (right panel). In each plot, the values of the parameter p (ranging from N to 1 000 in such a way that 100 equidistant points are obtained) have been mapped to the vector $(1, \ldots, 100)$ (variable p mapped to variable rep, abscissa). Six points correspond to each value of the vector, the ordinates of which are the power of the six tests: the point is represented by a black circle in the case of PAGEo, a red rhombus for PAGEr, a green triangle for PAGEp, a blue circle for KS, a light blue square for WKS and a pink star for WKSr.

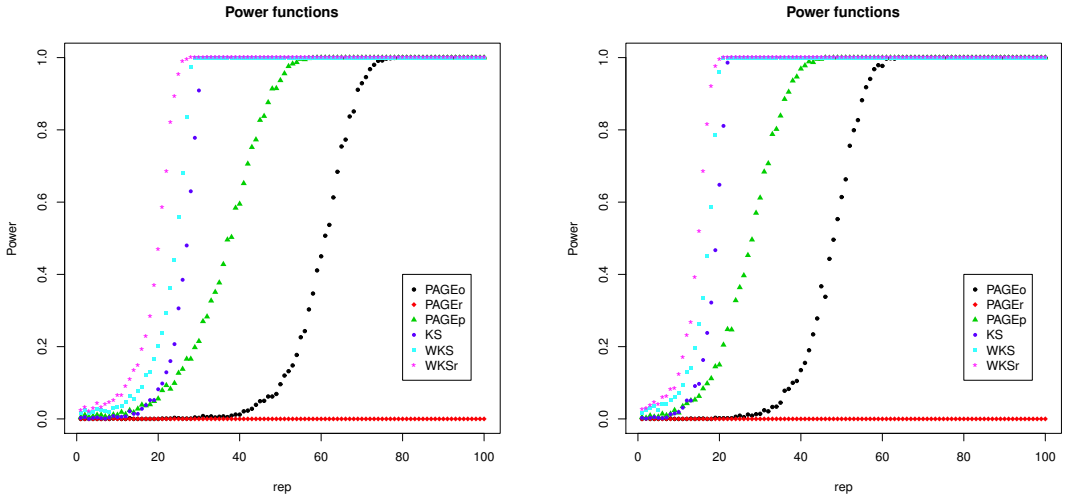


Figure 4.13: Power functions of the tests PAGEo, PAGEr, PAGEp, KS, WKS and WKSr for the third parametered family and sample size $n = 100$ (left panel). Power functions of the same tests for the third parametered family and sample size $n = 200$ (right panel). In each plot, the values of the parameter p (ranging from N to 1 000 in such a way that 100 equidistant points are obtained) have been mapped to the vector $(1, \ldots, 100)$ (variable p mapped to variable rep, abscissa). Six points correspond to each value of rep, the ordinates of which are the power of the six tests: the point is represented by a black circle in the case of PAGEo, a red rhombus for PAGEr, a green triangle for PAGEp, a blue circle for KS, a light blue square for WKS and a pink star for WKSr.

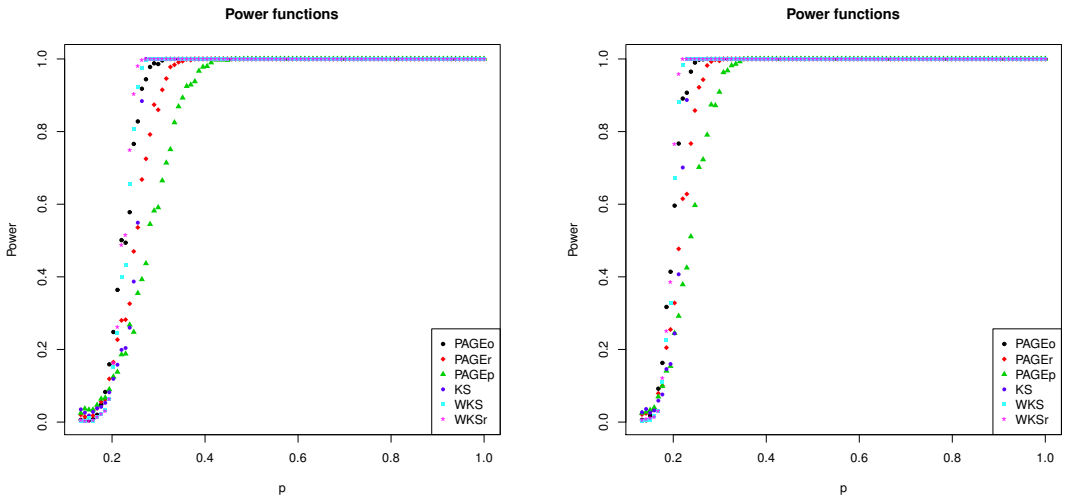


Figure 4.14: Power functions of the tests PAGEo, PAGEr, PAGEp, KS, WKS and WKSr for the fourth parametered family and sample size $n = 100$ (left panel). Power functions of the same tests for the fourth parametered family and sample size $n = 200$ (right panel). In each plot, the parameter p (abscissa) ranges from p_0 to 1 in such a way that 100 equidistant points are obtained. Six points correspond to each p, the ordinates of which are the power of the six tests: the point is represented by a black circle in the case of PAGEo, a red rhombus for PAGEr, a green triangle for PAGEp, a blue circle for KS, a light blue square for WKS and a pink star for WKSr.

Next, the same procedure was repeated for the DWKS test that was applied to the 4 parametered families above. The distribution function F was calculated each time from the 1 000 samples drawn according to $\mathcal{H}_p$ in the way described in subsection 3.2.4 for 10 groups of genes (see explanation for this choice in subsection 4.4.2). Figure 4.15 shows the results. We see that in all cases, the proportion of gene sets detected as significant is around 0.05- proportion when a test is repeated under its null hypothesis. In fact, the alternative hypotheses according to which the samples are drawn are very close to the null hypothesis of the DWKS test. More specifically, in the first parametered family the gene $\frac{i}{N}$ is assigned the probability $(1-p)+pg(\frac{i}{N})$ and then samples are drawn according to the Wallenius model. In the second parametered family, we have two groups of genes: the first p genes compose the first group and the rest $N-p$ genes the second. The inclusion probabilities are $\frac{1}{p}$ and 0 respectively. In the third parametered family, we also have two groups of genes: the first and the last p genes compose the first group and the rest $N-2p$ genes the second. The inclusion probabilities are $\frac{1}{2p}$ and 0 respectively. Of course, the further limitation that exactly half of the genes inside the gene set belong to the first p genes and the rest half to the last p genes has been imposed. Finally, in the fourth parametered family we have again two groups of genes: the "interesting" genes (number p_0N) and the "not interesting" genes (number $(1-p_0)N$). The inclusion probabilities p_1 and p_2 for the two groups should satisfy the conditions $\frac{p_1}{p_2}=\frac{p(1-p_0)}{p_0(1-p)}$ and $p_1p_0N+p_2(1-p_0)N=1$. Once again, the further constraint that the proportion of interesting genes inside the gene set be exactly p has been imposed. It is also worth noting that despite the fact that we make the estimation of the distribution F from a larger number of groups (10) than the real ones (2), the DWKS test seems to perform well. The reason is that the data are divided into more groups but the assigned inclusion probabilities are very close.

4.4 Curbing false detection rates

Most statistical tests assume as a null hypothesis that the gene set is a random sample without replacement from the total gene population. In this section, the problems that may arise, if this null hypothesis is used, are described. The correction brought by the DWKS test is shown after.

4.4.1 The choice of a null hypothesis

In this subsection, vectors of numeric data are compared to databases, considering the following 3 conditions:

- A: Actual vector of numeric data or database. The vector and the database are "reduced" to common gene symbols.
- E: Equal probability sampling without replacement. For the vector of numeric data, the numeric values are kept but they are indexed by a random ordering of the genes. For the database, simulated gene sets are drawn with the same sizes as in the actual database.
- U: Unequal probability sampling without replacement. For the vector of numeric data, the numeric values (sorted in decreasing order) are kept but they are indexed in a way which tends to rank the most frequent genes in the database first. The Wallenius model has been used: more specifically, all genes are assigned a probability proportional to their frequency in the database. Then, the genes are drawn one by one, in such a way that the probability of taking a particular gene at a particular draw is proportional to this gene's fraction of the total frequency of all genes that have not been drawn so far. For the database, samples are drawn without replacement, with probabilities proportional to the frequencies of genes in the database according to the Wallenius model too.

Applying those conditions to the vector of numeric data and the database yields 9 different (vector/database) pairs. Firstly, the C2 database was considered as actual database and a sample of expression levels from the GEO dataset GSE36133 [10] as actual vector. The two-sided WKS test was

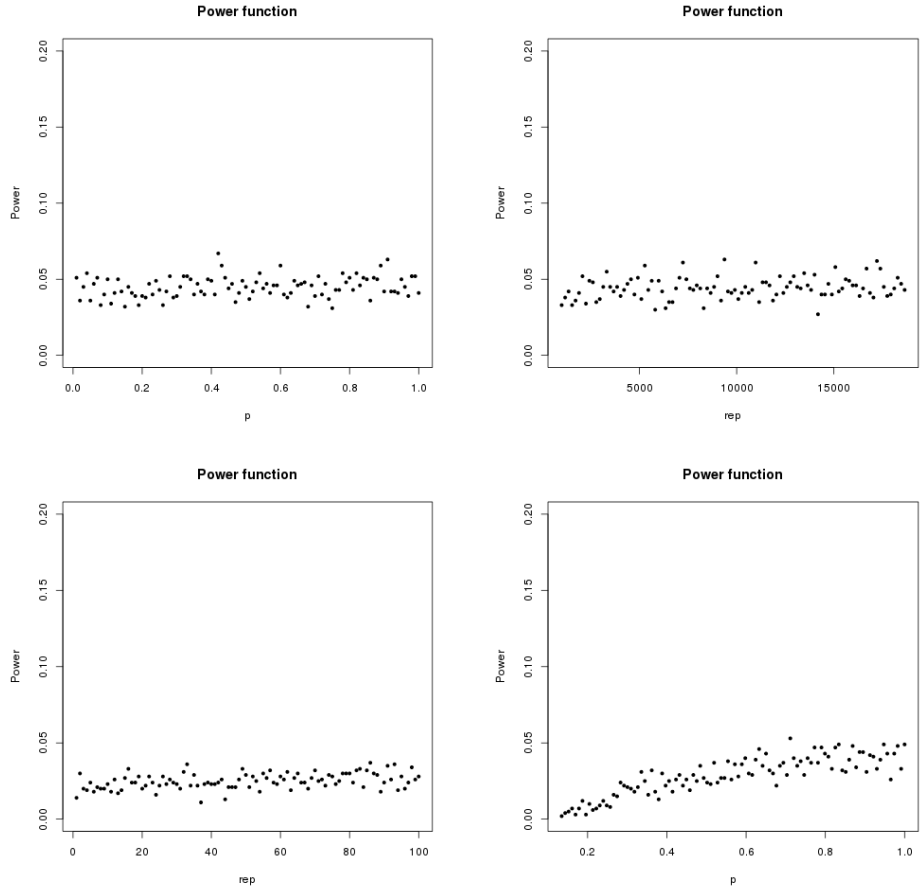


Figure 4.15: Power functions from application of the DWKS test to the four parametered families and sample size $n = 100$. The variable on the x-axis is the same as in Figures 4.10, 4.12, 4.13 and 4.14 respectively. For each x the ordinate is the power of the DWKS test.

applied to those 9 pairs. The results are depicted on Figure 4.16. Figure 4.16 shows "pvplots" [127]: such a graph plots the ordered p-values on $-\log_{10}$ scale.

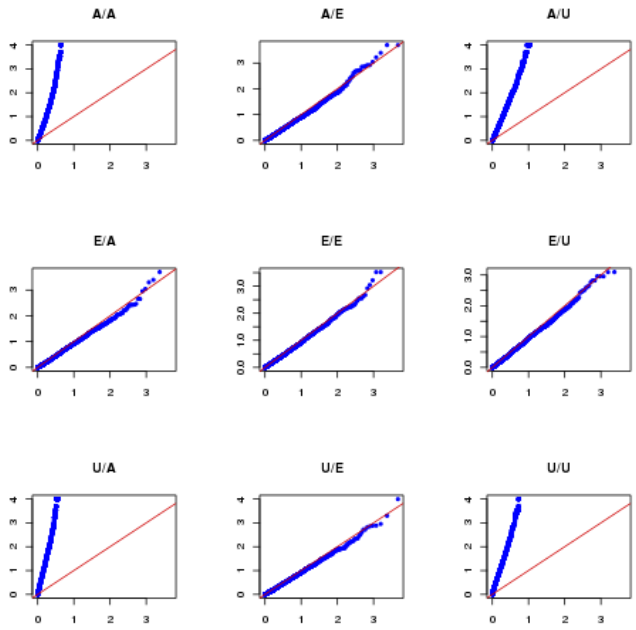


Figure 4.16: Pvplots for two-sided WKS under 9 different (vector/database) pairs for weights coming from real data. The ordered p-values (y-axis) are plotted against the corresponding quantiles of the uniform distribution in $-\log_{10}$ scale both. The pair under which the WKS test has been implemented is depicted in the title of each graph. As actual data, the C2 database and a sample of expression levels from the GEO dataset GSE36133 have been used.

Since there is lack of biological information in 8 out of the 9 pairs, the corresponding p-values should be uniformly distributed. However, this is true only in five pairs: A/E, E/A, E/E, E/U and U/E, where the first letter refers to the condition for the vector and the second to the condition for the database. The pairs A/E, E/E and U/E correspond to the null hypothesis of the WKS test. For the pairs E/A and E/U, the uniform distribution is still valid. On the contrary, in the three pairs A/U, U/U and U/A, significance is clearly exaggerated: the pvplots remain way above the bisector. We will explain the result in the U/U case: the indexing tends to rank the more frequent genes first. Yet, many gene sets contain the frequent genes. As a consequence, many gene sets will have many genes in the first part of the list and thus the cumulated weight function will be biased towards the left, it will be far from G and the gene set will end up being declared significant. Next, the values in the vector were replaced by their ranks and the procedure was repeated. The results are shown on Figure 4.17 (left panel). In addition, the one-sided WKS test (testing for enrichment) was applied to the initial expression vector and the C2 database (see Figure 4.17 (right panel)). The conclusions drawn are the same.

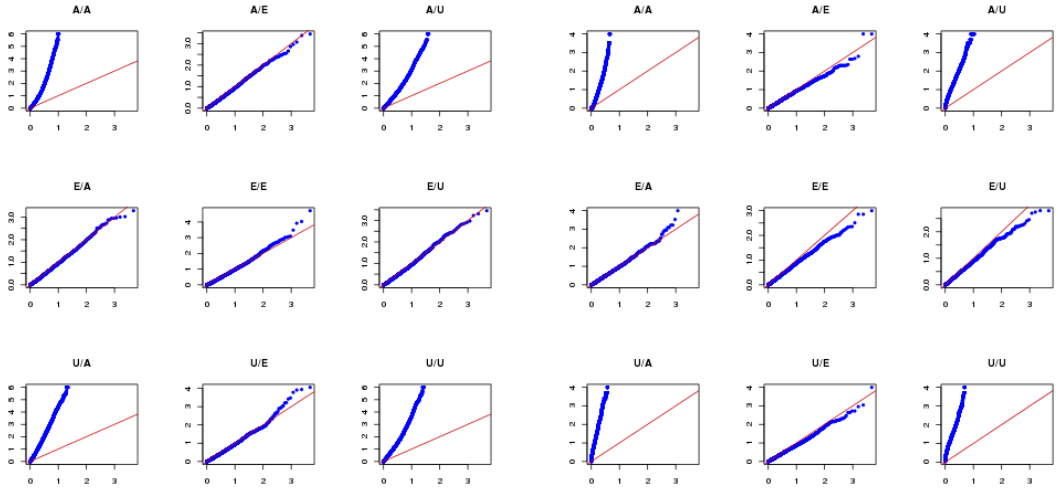


Figure 4.17: Pvplots for two-sided WKS under 9 different (vector/database) pairs for $g(x) = 2(1-x)$ (left panel). Pvplots for one-sided WKS under 9 different (vector/database) pairs for weights coming from real data (right panel). In both panels, the ordered p-values (y-axis) are plotted against the corresponding quantiles of the uniform distribution in $-\log_{10}$ scale both. The pair under which the WKS test has been implemented is depicted in the title of each graph. As actual data, the C2 database and a sample of expression levels from the GEO dataset GSE36133 have been used.

For comparison sake, the procedure was repeated for another statistical test, PAGE. The results are presented on Figure 4.18. The same conclusions can be drawn, that are also consistent with what is observed with Fisher's exact test [128].

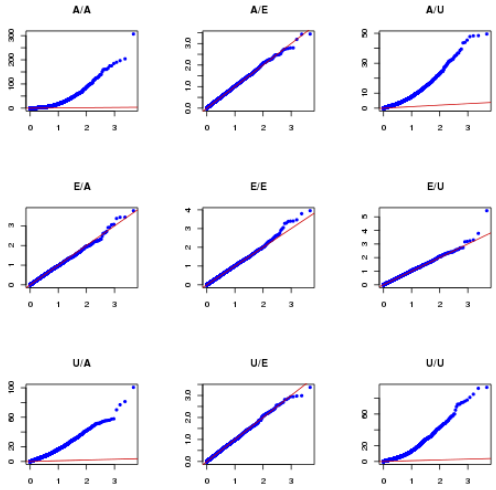


Figure 4.18: Pvplots for PAGE under 9 different (vector/database) pairs for weights coming from real data. The ordered p-values (y-axis) are plotted against the corresponding quantiles of the uniform distribution in $-\log_{10}$ scale both. The pair under which PAGE has been implemented is depicted in the title of each graph. As actual data, the C2 database and a sample of expression levels from the GEO dataset GSE36133 have been used.

4.4.2 Correction brought by the DWKS test

One issue concerning the DWKS test is the choice of the number of groups. The choice will be discussed based on the effect of different groupings in the application of the DWKS test. The following experiment was implemented. A vector and a database were simulated both according to the condition U described in the previous subsection. The actual data were one sample of expression levels from the GEO dataset GSE36133 and the C2 database. Then, the distribution function F was estimated from the simulated data for different numbers of groups in the way described in subsection 3.2.4 and the one-sided DWKS test (testing for enrichment) was applied. The number of groups was increased from 2 to 10 by step 2. All the pvplots so obtained are superposed on Figure 4.19.

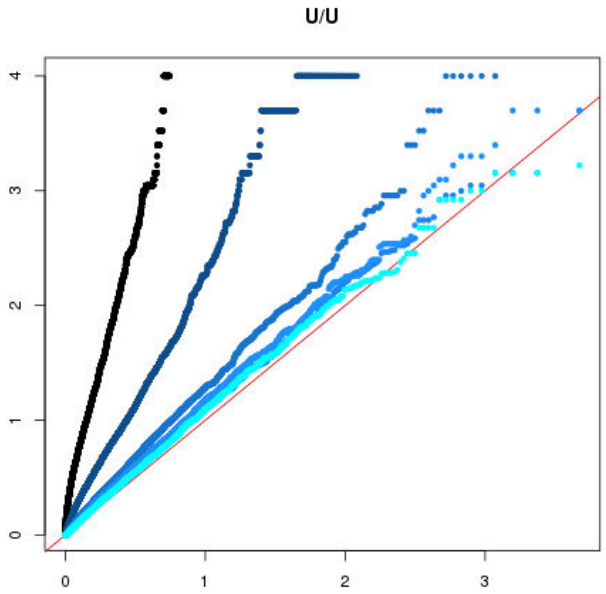


Figure 4.19: Pvplots for the one-sided DWKS applied to the (vector/database) pair U/U. The distribution function F was estimated for different numbers of groups. The pvplots for 2, 4, 6, 8 and 10 groups appear in increasingly lighter shades of blue. The pvplot for the one-sided WKS test appears in black.

From Figure 4.19, we see that the pvplots come closer to the theoretical red line as the number of groups increases. For $k = 10$ groups, an optimal fit is achieved. However, it should be noted that increasing further the number of groups does not improve the quality of the DWKS test, since from one point after the pvplots become indistinguishable.

Next, the one-sided DWKS test (testing for enrichment) was applied to the same 9 (vector/database) pairs, as these are described in subsection 4.4.1. The results are shown on the left panel of Figure 4.20. The same procedure was repeated with $g(x) = 1 - F(x)$ (see end of subsection 3.2.2). The pvplots so obtained are depicted on the right panel of Figure 4.20. For the estimation of the distribution function F, 10 groups were used in all cases for the reason explained above.

From Figure 4.20, we see that the p-values are uniformly distributed- as they should be- in 7 out of the 8 pairs of lack of biological information. The DWKS test succeeds to give uniformly distributed p-values for the most unrelevant with biological information pair U/U. It has also corrected the A/U case. However, the p-values in the U/A case, although less extreme than those produced by the WKS test, still remain above the bisector. One reason could be that the model for the database, that of gene sets where the genes are included with different probabilities, might not be the most appropriate. Another reason could be that the gene sets are not independent, since some groups of

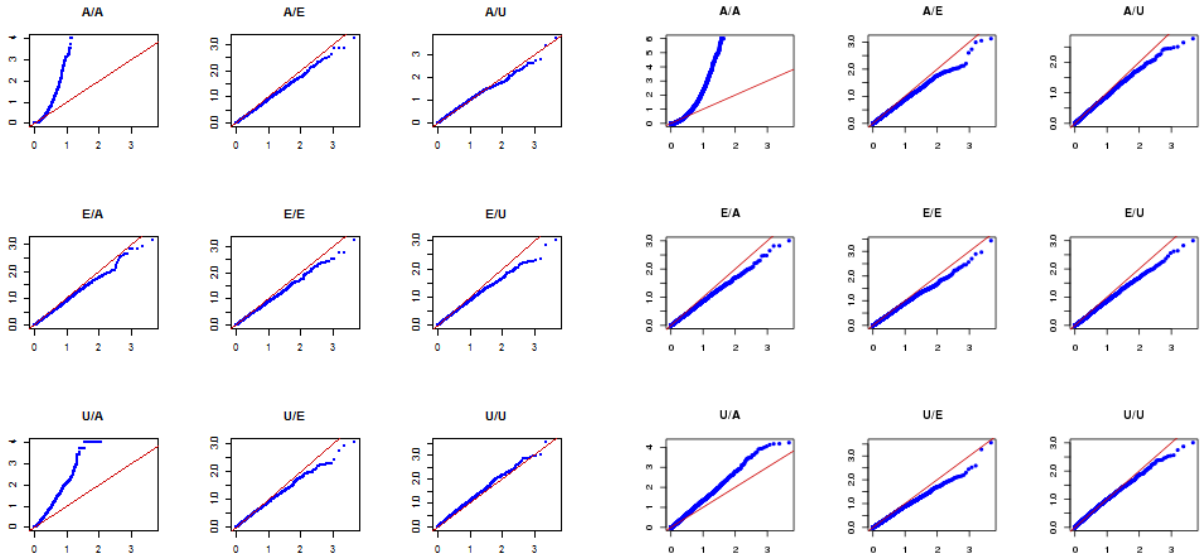


Figure 4.20: Pvplots for the one-sided DWKS under 9 different (vector/database) pairs for weights coming from real data (let panel). Pvplots for the one-sided DWKS under 9 different (vector/database) pairs for $g(x) = 1 - F(x)$ (right panel). The pair under which the DWKS test has been implemented is depicted in the title of each graph. As actual data, the C2 database and a sample of expression levels from the GEO dataset GSE36133 have been used.

genes systematically appear in many of those and thus the p-values so obtained are not independent either. Finally, the non-uniformity of p-values could be due to the inter-gene correlations.

4.4.3 Examples of distribution functions based on simulated and real data

The distribution functions F that were used for the DWKS test in each of the 9 different (vector/database) pairs of Figure 4.20 (left panel), as these were calculated in the way described in subsection 3.2.4, are depicted on Figure 4.21.

As expected, when either the database or the vector has been simulated according to the condition E, then the distribution function F is uniform. On the contrary, the estimated distribution function F corresponding to the pairs A/A, A/U, U/A and U/U is biased towards the left, due to the fact that the frequent genes in the database tend to appear first in the ranked vectors. Thus, once again, we see that the issue of unequal gene frequencies must be taken into account.

4.5 Application of the (D)WKS test to real data

In this section, a comparison between GSEA and the (D)WKS test is provided. A limited selection of real data is presented. Some of these can be found in [26]. However, we consider the results reported below as representative of the observations that can be obtained on different situations.

Several vectors coming from the GEO repository were tested against all 4 722 gene sets in the MSig C2 database, using the classical GSEA, and the WKS tests. The vectors that were used came from GEO dataset GSE36133 of [10], annotated using the org.Hs.eg.db package of [23]. This gave $N = 18\,638$ different gene names. Observe that applying the tests, the gene sets are necessarily reduced to those N genes. Out of the 21 047 different gene symbols present in C2, only 16 683 were common with the N genes of the chosen vectors.

For a given vector, two sets of 4 722 p-values were obtained, one with the GSEA test, the other

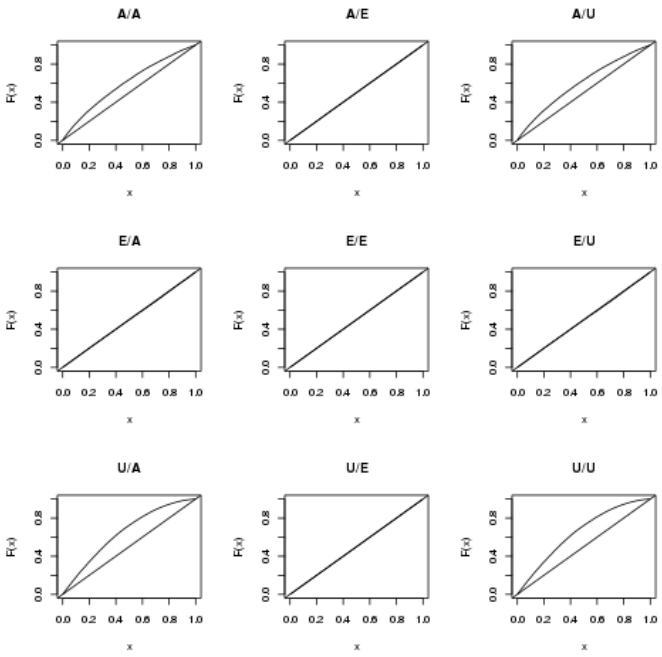


Figure 4.21: Distribution functions F used for the DWKS test in each of the 9 different (vector/database) pairs of the left panel of Figure 4.20.

with the WKS test. Results that can be considered as typical are represented on Figure 4.22. In that case, the vector contained expression data from liver tumor tissue. Out of the 4 722 gene sets of C2, 129 have "liver" in their title. They were considered as related to liver cancer, and the corresponding points are represented as red triangles on the figure. The negative logarithms in base 10 of the p-values of both tests have been plotted, thus the figure displays 4 722 points corresponding to p-value pairs. For comparison sake, only raw p-values are considered, without FDR adjustment. A p-value of 5% is marked by a dashed black line: points on the right of the vertical line are significant for the classical GSEA test, points above the horizontal line are significant for the WKS test. For the WKS test, the CDF F was calculated over $m = N = 18\,638$ discretization points, and the number of Monte-Carlo simulations was $nsim = 10^5$. For the classical GSEA test, the number of Monte-Carlo simulation had to be limited to 10^4.

The vertical dotted lines appearing on the right of the graphic are artefacts, due to the Monte-Carlo method for the GSEA test: the rightmost line corresponds to cases where the point-estimated p-value is equal to 0. Apart from these artefacts, it must be observed that the results of both tests are globally coherent: 2 501 database gene sets were significant (p-value smaller than 5%) for the WKS test, 2 764 for GSEA, 2 268 for both. There are no points in the bottom right corner of the graphics: when a p-value is very small for GSEA, it is never large for WKS. The converse is not true: many points in the upper left corner correspond to gene sets with a large p-value for GSEA, small for WKS.

More interesting is the analysis of liver-related gene sets. Out of 129, 76 were declared significant by the WKS test; 70 by the GSEA test, 66 by both. Therefore, 10 gene sets were declared significant by WKS only, and 4 by GSEA only. Figure 4.23 plots the cumulated proportions of weights $S_n(t)$ for those 14 gene sets. On the same plot, the functions t (bisector), to which the classical GSEA test compares $S_n(t)$, and $G(t)$, used as a centering by WKS, also appear. On the graphic, the reason why a gene set may be declared significant by one test and not the other, is clear. The 4 gene sets declared significant by GSEA and not WKS, are represented by blue step functions; they are above the G curve. They are indeed far from the bisector, but not far enough from G. Inside the corresponding gene sets, the weights of the genes tend to be representative of the global distribution of weights, and

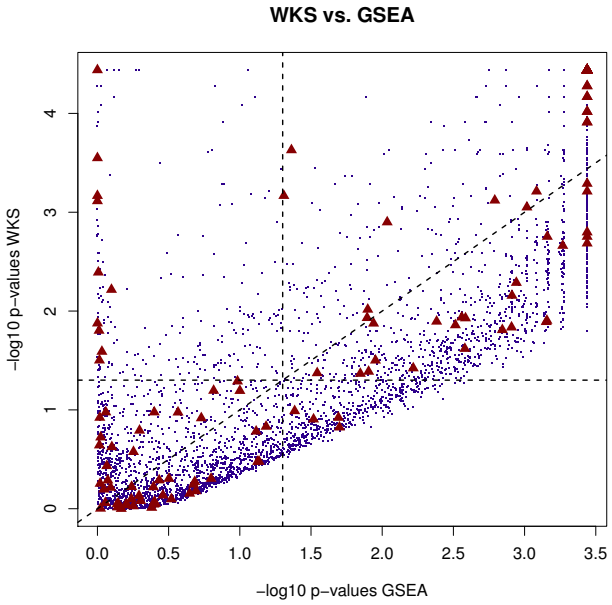


Figure 4.22: Test of a liver tumor expression vector against the 4 722 gene sets of the MSig C2 database. Each point corresponds to a gene set, the coordinates being the negative logarithm in base 10 of the p-values, for the classical GSEA and the WKS tests. Gene sets related to liver cancer in the database are represented as red triangles. The horizontal and vertical dashed lines correspond to 5% p-values.

Figure 4.23: Plots of the cumulated weight function $S_n(t)$ for vectors declared significant by WKS and not GSEA (red step functions) and conversely (blue step functions). The functions t (to which the classical GSEA test compares $S_n(t)$), and $G(t)$ (used as a centering by WKS), are dashed.

declaring them as significant by comparing to the bisector can be regarded as a bias. Moreover, it should be observed that 3 out of the 4 have size below 19. As already explained in subsection 3.3.1, when dealing with very small sizes, the WKS test tends to underrate significance.

Conversely, the 10 gene sets declared significant by WKS and not GSEA are represented by red step functions. They are relatively close to the bisector as expected, but clearly below the G curve, to which WKS compares. This means that in the corresponding gene sets, the genes tend to have significantly smaller weights, i.e. they are significantly underexpressed. An interesting example is the gene set named `Acevedo_methylated_in_liver_cancer_dn`. As indicated by the two letters `dn`, it contains genes which are known to be down-regulated in case of liver cancer [1]. On Figure 4.22, it appears on the upper left corner: it has p-value close to 0 for WKS, close to 1 for GSEA. Thus WKS has detected it as significantly related to the tested vector, whereas GSEA has not. The case is not unique: 3 gene sets had p-value larger than 0.5 for GSEA, smaller than 10^{-3} for WKS.

As already stated, these results were consistently observed for different expression vectors, from different types of cancers. In all cases, WKS declared less significant gene sets than GSEA in a proportion of about 10% from the whole database, whereas it tended to detect more significant gene sets among those related to the correct type of cancer.

Next, the GEO dataset GSE3165 of [56] was studied. An analysis, similar to the one conducted by [121], has been carried out. The dataset was annotated using Carlson's package [22], which yielded $N = 12\,747$ different gene names, since only the platform GPL887 was considered. Subsequently, the 94 samples were divided into two groups: those with basal-like breast cancer and those with one of the other five subtypes of cancer (luminal A, luminal B, Her2, normal-like and claudin-low). The row means were computed over the two submatrices yielding two vectors of size 12 747. The difference between these two vectors was then calculated, giving a vector containing the log(FC) between basal-like and non-basal tumors. This vector was tested against the C2 database, using the classical GSEA, and the WKS tests.

The results are shown on Figure 4.24. Out of the 4 722 gene sets, 1 229 were declared significant by GSEA and 835 by WKS. Out of the 4 722 gene sets, 159 have "breast" in their title. Regarding the breast-related gene sets, 89 were detected as significant by GSEA, 80 by WKS and 79 by both. Thus, 1 gene set is declared significant by WKS only, and 10 by GSEA only. Although GSEA detects more breast-related gene sets, the WKS test still outperforms GSEA, in the sense that the proportion of the breast-related gene sets declared significant over the total number of gene sets declared significant remains higher. Figure 4.25 plots the cumulated proportions of weights $S_n(t)$ for those 11 gene sets, together with the bisector and the function G. Regarding the gene sets declared significant by GSEA only, 7 out of 10 have size below 39 and there, the WKS test tends to underrate significance. Regarding the remaining 3 gene sets, they do not lie very far from the G curve. On the contrary, from the right panel, we see a gene set with clear enrichment at the top of the list that GSEA fails to detect. Another representation for the 11 gene sets, similar to the one provided in [108], is given in Figures 4.26 and 4.27. In this representation, the difference $S_n(\cdot) - G(\cdot)$ is plotted, while at the bottom of each graph vertical lines are added, indicating the positions of the genes inside the gene set in the ranked list. These graphical plots help biologists interpret the results more easily, since they can see immediately the distribution of the gene set in the ranked list.

To validate further the fact that the WKS test can be really informative from a biological perspective and also evaluate the performance of the one-sided WKS test, the vector of the $|\log(\text{FC})|$ was also considered. Then, the one-sided WKS test (corresponding to the test statistic $\sup(S_n(t) - G(t))$) was applied. The 10 most significant gene sets so obtained, are summarized on Table 4.5.

From Table 4.5, we see that all gene sets, but the last one, are related to breast cancer. More importantly, the two basal-like signatures come first. On the contrary, the two basal-like signatures come in positions 20 and 25, when GSEA is applied on the same data.

The GSEA was applied to the same vector. The results are depicted on Figure 4.28. Out of the 4 722 gene sets, 1 430 were declared significant by GSEA and 659 by WKS. Regarding the breast-related gene sets, 93 were detected as significant by GSEA, 65 by WKS and 62 by both. Thus, 3 gene sets are declared significant by WKS only, and 31 by GSEA only. Figure 4.29 plots the cumulated proportions of weights $S_n(t)$ for those 34 gene sets, together with the bisector and the function G.

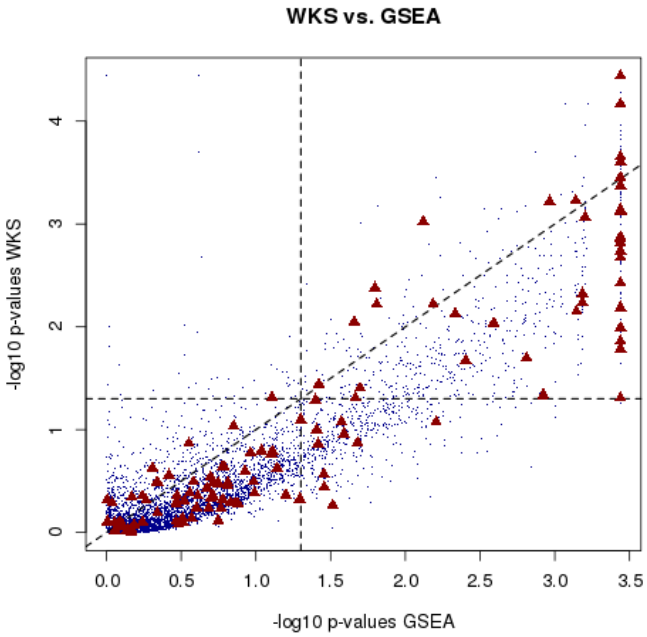


Figure 4.24: Test of a vector of log(FC) derived from the GEO dataset GSE3165 against the 4 722 gene sets of the MSig C2 database. Each point corresponds to a gene set, the coordinates being the negative logarithm in base 10 of the p-values, for the classical GSEA and the WKS tests. Gene sets related to breast cancer in the database are represented as red triangles. The horizontal and vertical dashed lines correspond to 5% p-values.

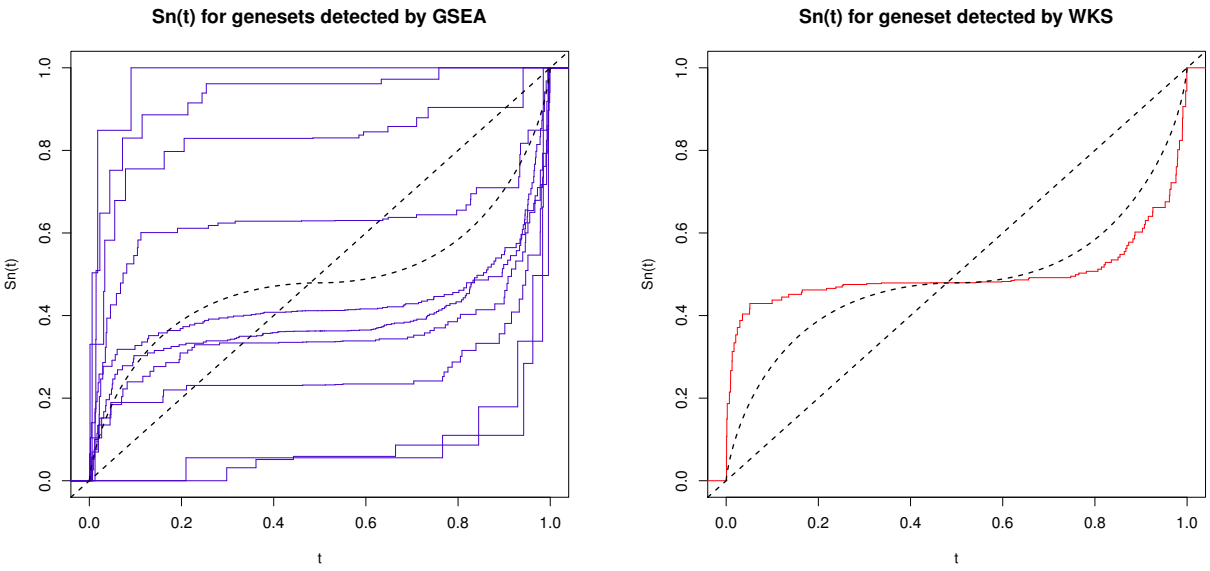


Figure 4.25: Plots of the cumulated weight function $S_n(t)$ for vectors declared significant by GSEA and not WKS (blue step functions, left panel) and conversely (red step function, right panel). The functions t (to which the classical GSEA test compares $S_n(t)$), and $G(t)$ (used as a centering by WKS), are dashed.

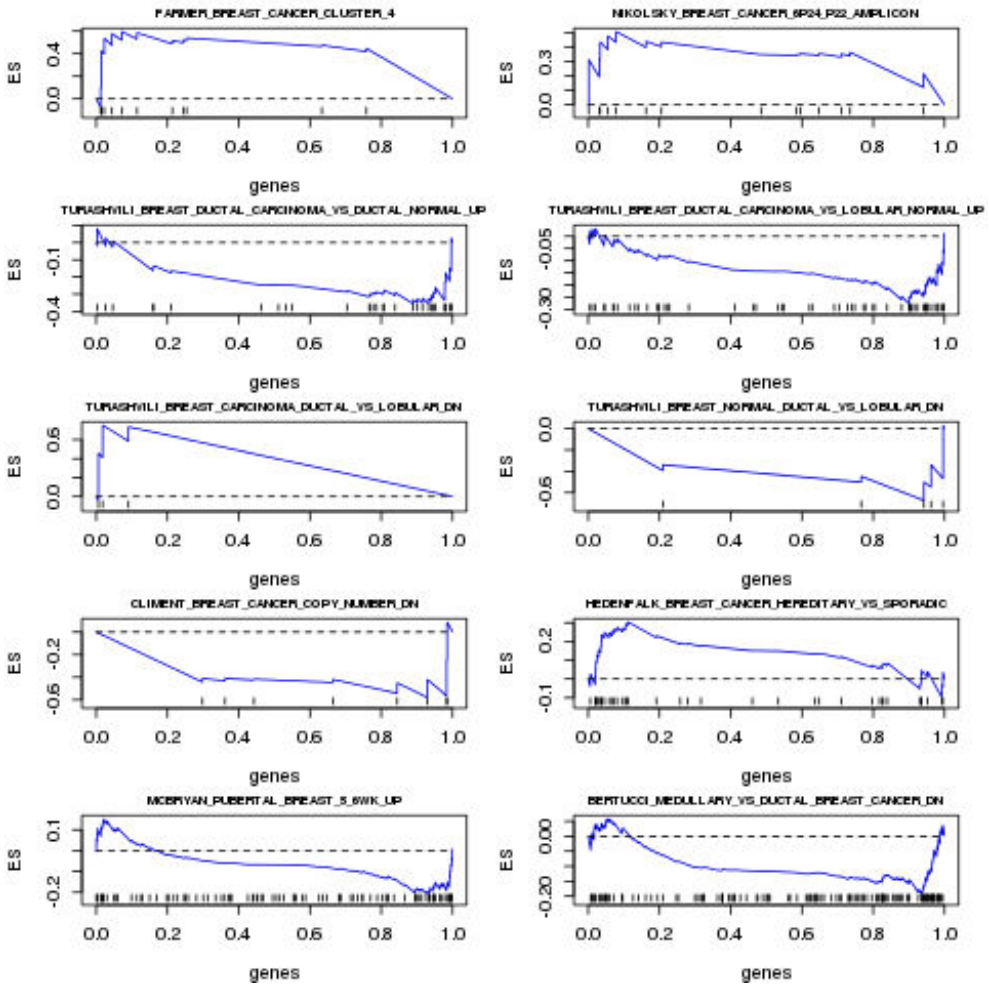


Figure 4.26: Plots of $S_n(\cdot) - G(\cdot)$ for vectors declared significant by GSEA and not WKS. At the bottom of each graph vertical lines are added, indicating the positions of the genes inside the gene set in the ranked list.

Gene sets
Smid_breast_cancer_basal_dn
Smid_breast_cancer_basal_up
Farmer_breast_cancer_basal_vs_luminal
Smid_breast_cancer_luminal_b_dn
Smid_breast_cancer_relapse_in_bone_dn
Gozgit_esr1_targets_dn
Charafe_breast_cancer_luminal_vs_basal_up
Doane_breast_cancer_esr1_up
Lien_breast_carcinoma_metaplastic_vs_ductal_dn
Meissner_brain_hcp_with_h3k4me3_and_h3k27me3

Table 4.5: Ten most significant gene sets declared by the WKS test, applied to the vector of $|\log(\text{FC})|$ derived from the GEO dataset GSE3165 and the C2 database.

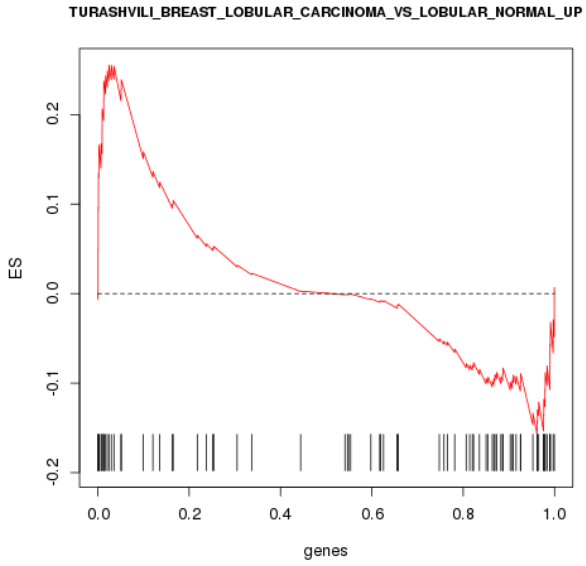


Figure 4.27: Plot of $S_n(\cdot) - G(\cdot)$ for the vector declared significant by WKS and not GSEA. At the bottom of the graph vertical lines are added, indicating the positions of the genes inside the gene set in the ranked list.

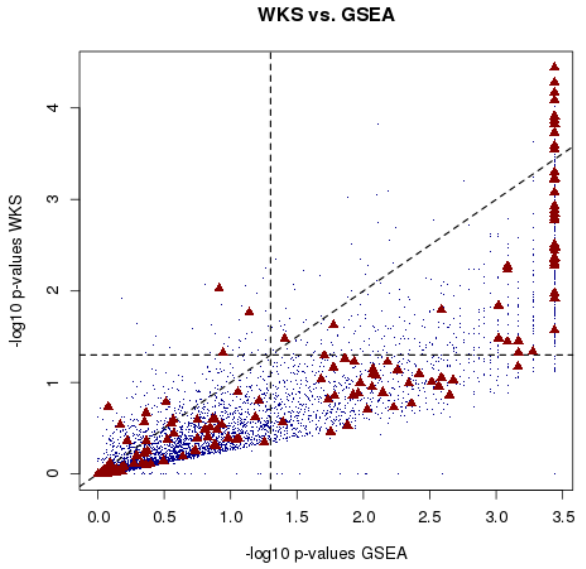


Figure 4.28: Test of a vector of $|\log(\mathrm{FC})|$ derived from the GEO dataset GSE3165 against the 4 722 gene sets of the MSig C2 database. Each point corresponds to a gene set, the coordinates being the negative logarithm in base 10 of the p-values, for the classical GSEA and the WKS tests. Gene sets related to breast cancer in the database are represented as red triangles. The horizontal and vertical dashed lines correspond to 5% p-values.

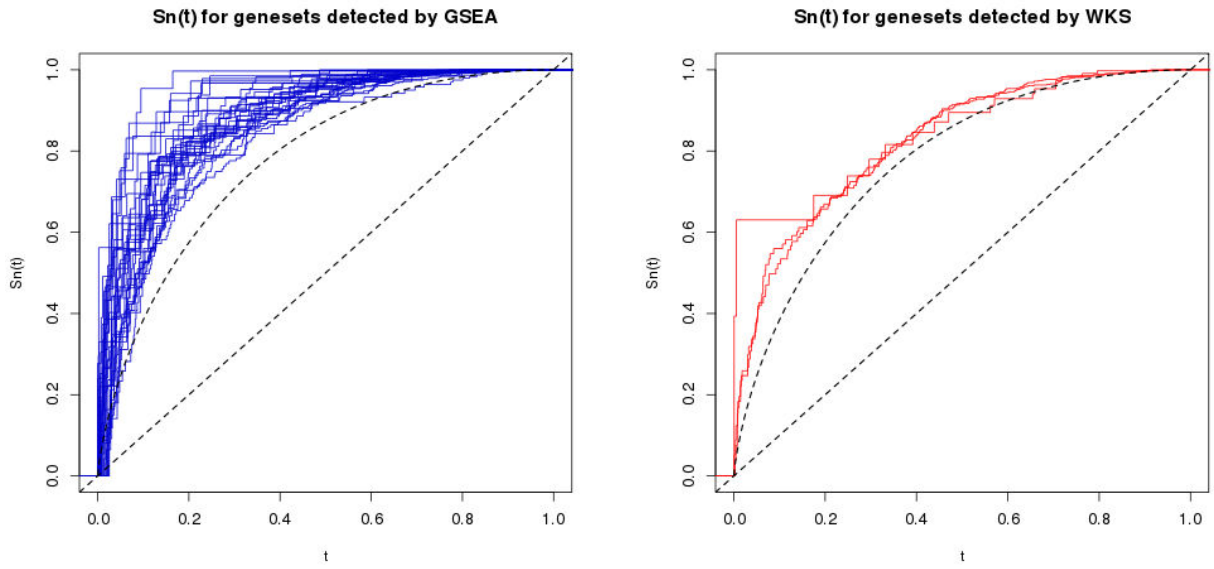


Figure 4.29: Plots of the cumulated weight function $S_n(t)$ for vectors declared significant by GSEA and not WKS (blue step functions, left panel) and conversely (red step function, right panel). The functions t (to which the classical GSEA test compares $S_n(t)$), and $G(t)$ (used as a centering by WKS), are dashed.

From the above analysis, we deduce that the "interesting" gene sets that GSEA fails to detect can be divided into two categories:

1. When the weights are positive, as in the first example, then GSEA fails to detect gene sets with a modest depletion. In Figure 4.30, we have plotted the GSEA null density for one of the 10 gene sets declared significant by WKS only (gene set named `Acevedo_methylated_in_liver_cancer_dn`), when analyzing the liver tumor expression vector, together with the observed test statistic. Figure 4.30 reveals why GSEA fails to detect it. Under the null hypothesis, the cumulated weight function $S_n(\cdot)$ will be close to the function G. Thus the GSEA test statistic for a random gene set will be close to $\sup_{t\in[0,1]} |\, G(t) - t \,|$. This is why the null density in Figure 4.30 is centered around the $\sup_{t\in[0,1]} |\, G(t) - t \,|$. On the contrary, the observed test statistic is very small, it does not even belong to the support of the distribution. However, since the p-values are right-tail probabilities, the p-value obtained is equal to one. Of course, when there is a very clearly marked depletion, then GSEA will detect it too.

2. When the weights are positive then GSEA fails to detect gene sets with a clear enrichment at the top of the list. When the weights take negative values as well (for instance log(FC)), GSEA might fail to detect gene sets with a clear enrichment at the top of the list or a clear depletion at the bottom of it. Figure 4.31 depicts the GSEA null density for the gene set of Figure 4.25 (right panel). To illustrate the first case in the second category, a gene set of size 155 was simulated in the following way: the gene set contains the 20 first genes of the liver tumor expression list and a sample of size 135 out of the genes $\{2\,000/18\,638, \ldots, 1\}$. We thus have a gene set with a clear enrichment at the top of the list and uniform distribution from one point after. The GSEA null density for a gene set of the same size has been computed. The results are shown on Figure 4.32. GSEA fails to detect such gene sets by assigning them a probability bigger than 0.05.

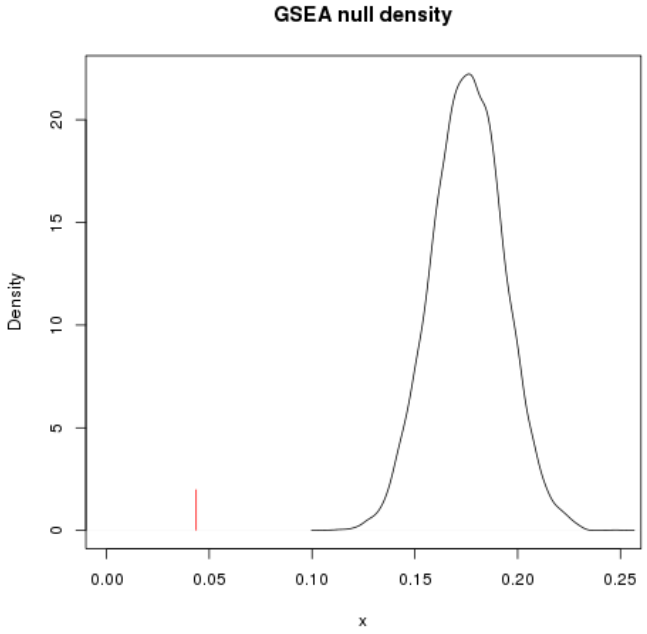


Figure 4.30: Null density of the GSEA test statistic for a gene set of size 699, where the liver tumor expression vector has been used. A red vertical line has been added at the value of the observed test statistic for the gene set named Acevedo_methylated_in_liver_cancer_dn.

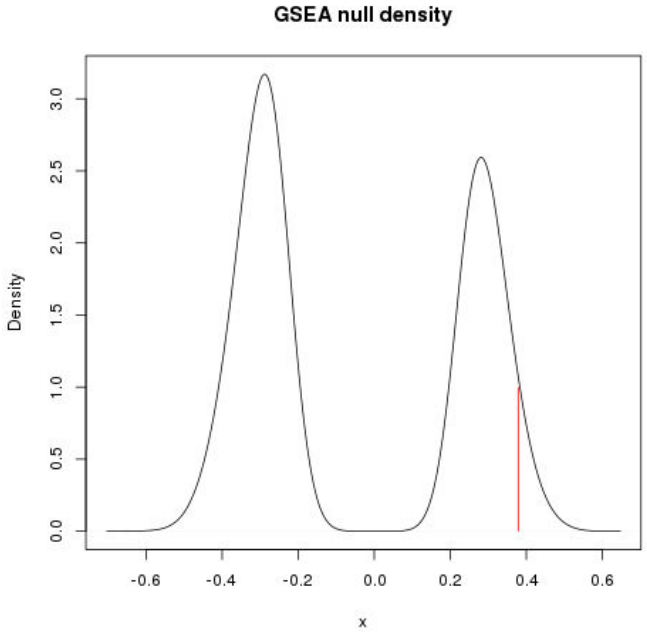


Figure 4.31: Null density of the GSEA test statistic for a gene set of size 77, where the vector of $|\log(FC)|$ has been used. A red vertical line has been added at the value of the observed test statistic for the gene set named Turashvili_breast_lobular_carcinoma_vs_lobular_normal_up.

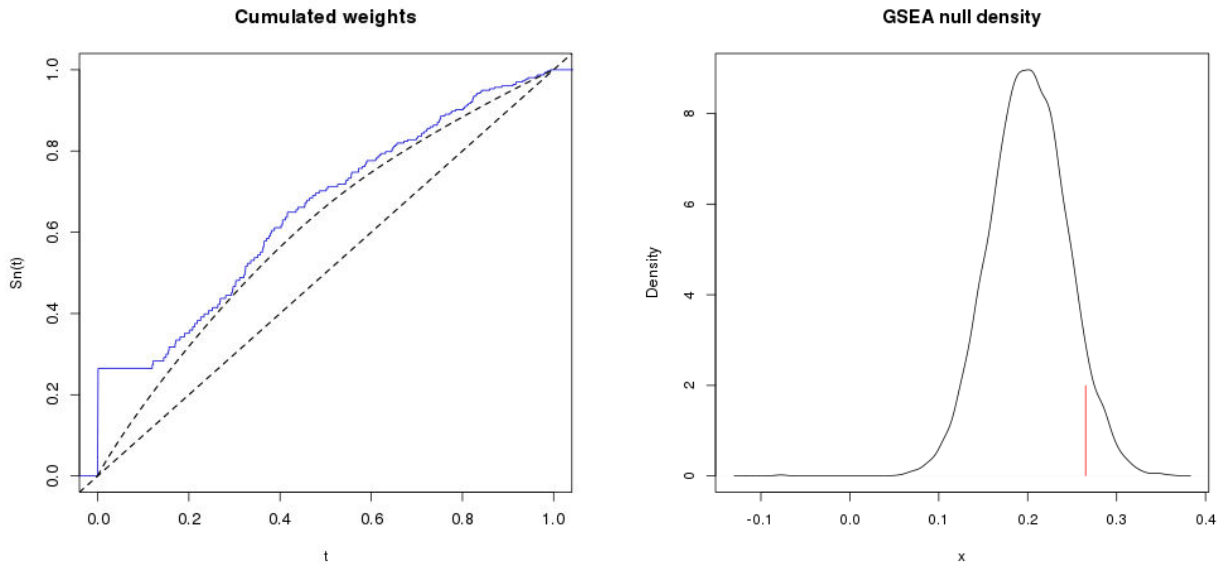


Figure 4.32: Plot of the cumulated weight function $S_n(t)$ for a simulated gene set with a clear enrichment at the top of the list (left panel). Null density of the GSEA test statistic for a gene set of the same size together with the observed test statistic (right panel). The liver tumor expression vector has been used.

Next, the vector of $|\log(\text{FC})|$ was tested against the C2 database, using the one-sided DWKS test. The distribution F was defined in the way described in subsection 3.2.4, where we used 10 groups. Out of the 4 722 gene sets, 315 were declared significant by the DWKS test. The 10 most significant gene sets so obtained, are reported on Table 4.6.

Gene sets
`Smid_breast_cancer_basal_dn`
`Farmer_breast_cancer_basal_vs_luminal`
`Smid_breast_cancer_basal_up`
`Smid_breast_cancer_luminal_b_dn`
`Doane_breast_cancer_esr1_up`
`Smid_breast_cancer_relapse_in_bone_dn`
`Lien_breast_carcinoma_metaplastic_vs_ductal_dn`
`Vantveer_breast_cancer_esr1_up`
`Charafe_breast_cancer_luminal_vs_basal_up`
`Gozgit_esr1_targets_dn`

Table 4.6: Ten most significant gene sets declared by the DWKS test, applied to the vector of $|\log(\text{FC})|$ derived from the GEO dataset GSE3165 and the C2 database.

As expected, the DWKS test reduces the number of gene sets declared significant: the number has been reduced by half compared to the WKS test. From Table 4.6, we see that the basal-like signatures remain in the first positions, while the non-relevant gene set (`Meissner_brain_hcp_with_h3k4me3_and_h3k27me3`) appears later.

Finally, applications of the KS (WKS with constant weights) test to real data are reported in [74].

4.6 AutoCompare software and R codes

In this section, the AutoCompare software that has been developed from our theoretical results is presented first. Next, instructions on how to use the R codes for data formatting and (D)WKS testing (given in the Appendix) are displayed.

4.6.1 AutoCompare_ZE-WKS software

The software tool AutoCompare_ZE-WKS was developed in an attempt to make it easier for experimental biologists to use the proposed tests. It consists an extension of the software tool AutoCompare, which implemented the FE test [92]. The tool implements the ZE test [128] (see subsection 2.3.3), the WKS test and the special case of it, WKSr for the moment. The DWKS test will be incorporated in the near future. The AutoCompare_ZE-WKS interface is shown on Figure 4.33.

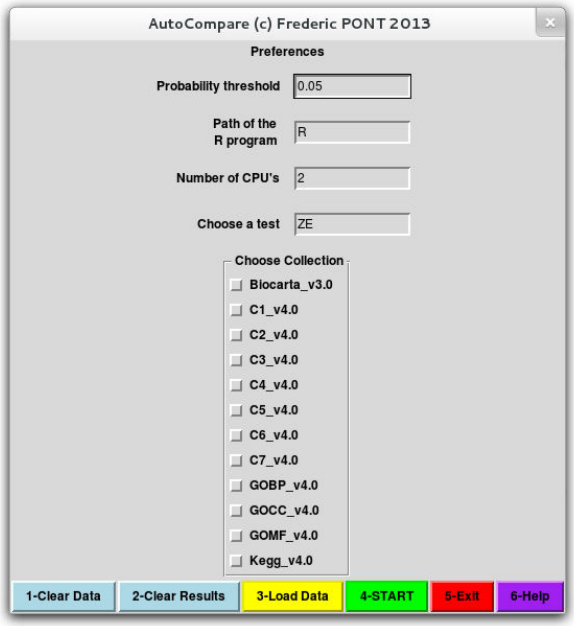


Figure 4.33: Screenshot of AutoCompare_ZE-WKS under linux

The software takes as input a data text file: either a file with one column containing the "interesting" gene symbols or a file with two columns, the first column containing all gene symbols and the second column the corresponding weights. For the second type of data, they do not have to be ordered. It performs the chosen test (one of ZE, WKS or WKSr) between the text file and the chosen database. A p-value is computed for every gene set of the database. The output file is a text file containing the gene sets declared significant from the above analysis together with the p-value. We provide AutoCompare_ZE-WKS with all MSigDb collections v4.0 (see subsection 2.1.3 and `http://www.broadinstitute.org/gsea/msigdb/collections.jsp#C2`). They amount to 10 295 different gene sets. However, the users can implement, in a very straightforward way their own gene sets (text files) or databases. It is worth noting that the user can choose more than one database and the tests will be made in parallel, divided in the different CPU's.

AutoCompare_ZE-WKS was developed using the Perl programming language (Perl v5.14.2, `http://www.perl.org/`) and the R statistical programming language [94] under the Linux operating system (Debian 7.7, `https://www.debian.org/`). AutoCompare_ZE-WKS is available for both Linux and Windows and runs on any operating system with Perl, either as a command line tool or with a graphical interface.

Just like AutoCompare [92], AutoCompare_ZE-WKS is very easy to use. Another advantage is that the calculations have been designed to be made in parallel. In this way, a text file can be tested against more than one databases at the same time, and the load of a serial computation is reduced according to the number of available CPU's.

In more detail, in order to run the application, the following are required:

– Perl version 10.0 or later

– R statistical language

– module Perl/Tk

– R package snowfall

The first three can be installed by following the instructions described in the autocompare_manual.pdf from `https://sites.google.com/site/fredsoftwares/products/data-mining`. Concerning the R package snowfall, open R and type
`install.packages("snowfall")`.

Unzip `AutoCompareZE_WKS.tgz`: this will create a directory `AutoCompareZE_WKS`. This directory contains:

– the 7 MSigDb collections, together with some subcollections, as subdirectories

– the subdirectories `data` and `results`

– the basic perl script `ZE_WKS.pl`, together with another perl script called by the first, and 3 R scripts implementing the statistical tests.

In order to start the application, if you work with linux, open a terminal and type `perl ZE_WKS.pl`, or double click on the file ZE_WKS.pl, if you work with windows. The AutoCompare_ZE-WKS window (see Figure 4.33 or 4.36) will open.

Choose software options:

- Probability threshold: The significance level. All gene sets with p-value beyond this level will be returned as significant. Default value: 0.05.
- Path of the R program: R is needed to perform the test. This line indicates where the R executable is installed. Default value: R.
- Number of CPU's: The test between a given list and different databases can be made in parallel. Define the number of CPU's you want to be used for the parallel computations. It should be a number between 1 and the number of machine's CPU's. Default value: 2.
- Choose a test: The test must be one of ZE, WKS or WKSr. If ZE is written, then the ZE test will be performed. If you want to apply the WKS test, then you should write WKS and finally, if you want to apply the WKS test with the actual weights being replaced by ranks, then you should write WKSr. Default value: ZE.

Then, you should choose the databases you want your list to be tested against. You can choose as many databases as you want. The databases will be divided in the CPU's successively, until all tests have been implemented.

The Clear Data button erases all the existing files in the subdirectory data of the current directory. The same operation is carried out for the subdirectory results, if you click on the Clear Results button.

The Load Data button helps you load the desired for testing text file. If you click on the Load Data button, then the window shown on Figure 4.34 opens. The text file can be stored anywhere on the computer. You just have to choose it and then by clicking on the Accept button, the file is automatically copied in the data directory.

To start the analysis, click on the START button. The file(s) in the data directory will be tested against the chosen databases, with the test that has been selected. The computations will be divided in as many CPU's as the number entered by the user. Once the analysis is over, a message will appear on the terminal informing you so:

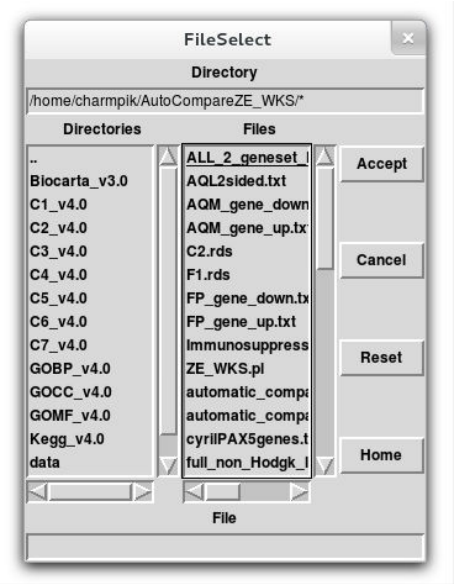


Figure 4.34: Window opening when clicking on the Load data button

```
     End of analysis !
=========================================================

The results directory contains all the details of the analysis.

Results files are not automatically erased, you have to remove them if not using
 the graphical interface.

=========================================================
```

The results are stored in the results directory. There will be one file with the significant gene sets (names and associated p-values) for every chosen database tested against the data file. Those gene sets with a p-value beyond the value entered for the parameter probability threshold will be declared significant. If the ZE test has been carried out, then the result files contain much more information: not only the gene sets declared significant with their associated p-values, but also the p-value from application of the FE test, the number of common genes with the input vector and finally the common gene symbols for each significant gene set are contained. The files can be easily identified, since their name is composed by the name of the data file and the name of the database used for testing. In the case of WKSr test, the extension "_rank" is further added (so that the file be distinguished from the corresponding file resulting from application of the WKS test between the same data file and database).

Clicking on the Help button will open a window (shown on Figure 4.35), displaying the basic instructions of use of the platform. Finally, you may click on the Exit button to close the application.

The above analysis concerned both operating systems: linux and windows. The only difference is that the AutoCompare_ZE-WKS window looks a little bit different under windows (see Figure 4.36).

4.6.2 Code for data formatting

In this subsection, instructions on how to use the R script `sagd.r` performing the downloading and formatting of the data exposed in subsection 2.2.1 are given first. The functions that have been encoded are displayed in the Appendix. It should be noted that the R script `sagd.r` contains many more functions that perform further statistical treatments. Here, only the part corresponding to the

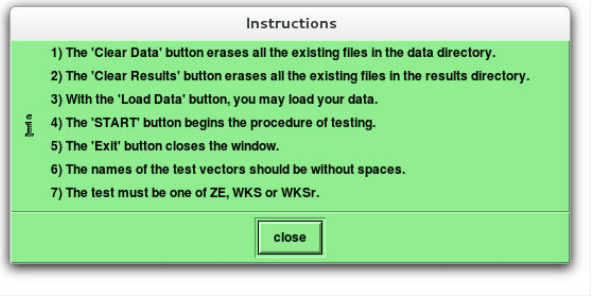


Figure 4.35: Window opening when clicking on the Help button

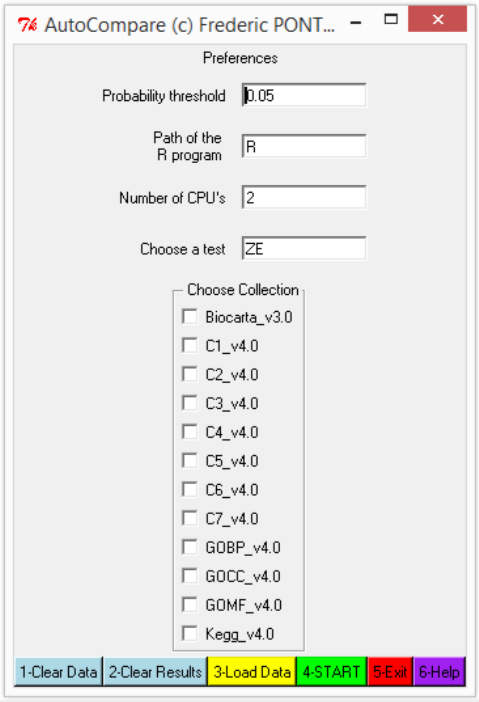


Figure 4.36: Screenshot of AutoCompare_ZE-WKS in windows

current work will be presented.

1. Download and install R from
   `http://www.r-project.org/`
2. download `sagd.tgz` from
   `http://ljk.imag.fr/membres/Bernard.Ycart/publis/sagd.tgz`,
   save it to your disk, and unzip it
3. source the script: `source("sagd.r")`
4. declare the working directory in which you want to save and read data matrices: something like
   `setwd("E:/mypath/mydir/gse_data/")`

The function for downloading and formatting data is `read.data`. Its variable is a GSE number: see `http://www.ncbi.nlm.nih.gov/geo/` and subsection 2.2.1. For instance GSE13159 is the Microarray

Innovations in LEukemia (MILE) study.
Type `read.data(13159)`, then return. This will download the data from the GEO website, format them, and save them as two files in your working directory: `GSE13159.rds` and `GSE13159_Info.rds`. This operation can be batched over a vector of GSE numbers. Here are the commands that download the 17 matrices of section 4.2 (may take several hours, and induce memory problems for the biggest matrices: make sure you have enough memory available).

```
vgse <- {c(2658,6891,7307,15061,20142,24080,31312,
33828,36133,36192,36382,36809,37069,39582,48348,48433,48762)}
read.data(vgse)
```

4.6.3 The WKS codes

In this subsection, instructions on how to use the R script `wks.r` performing the WKS and GSEA tests and providing with different visualizations of enrichment scores and p-values are given first. The functions that have been encoded are displayed in the Appendix. The same procedure is followed for the DWKS test after.

1. Download and install R from
   `http://www.r-project.org/`
2. download `wks.tgz` from
   `http://ljk.imag.fr/membres/Bernard.Ycart/publis/wks.tgz`,
   save it to your disk, and unzip it

The file `wks.tgz` unzips into a directory `/wks/` which contains:

- `wks.r`: the script file. Open R, set `/wks/` as working directory, then source the script: `source("wks.r")`.
- `C2.rds`: the MSig database C2 from [108]. It is encoded as a list of vectors of character strings: see [125]. Each vector in the list is a gene set. Load the database by:

  ```
  C2 <- readRDS("C2.rds")
  ```

- `F1.rds`, `F1os.rds`: pre-calculated distribution function for WKS test applied to rank statistics;
- `breast.rds`, `kidney.rds`, `liver.rds`: three sample vectors of expression levels, from the GEO dataset GSE36133 of [10]:
  GSM887033 (breast), GSM886844 (kidney), GSM887630 (liver). Load them by:

  ```
  b <- readRDS("breast.rds")
  k <- readRDS("kidney.rds")
  l <- readRDS("liver.rds")
  ```

Two test functions are provided: `WKS.test` and `GSEA.test`. Both take a vector and a database as their first two entries. Default values are provided for the other variables. The output is a sorted vector of p-values, indexed by the gene set names. The three alternatives are `"greater"`, `"less"`, and `"two.sided"` (default). For `"alternative="greater"`, the more enriched the gene set, the smaller the p-value. Respectively, for `"alternative="less"`, the more depleted the gene set, the smaller the p-value.

```
                                # default: two.sided
pvb <- WKS.test(b,C2)
pvk <- WKS.test(k,C2)
pvl <- WKS.test(l,C2)
```

```
                                # enriched
pvbg <- WKS.test(b,C2,alternative="greater")
pvkg <- WKS.test(k,C2,alternative="greater")
pvlg <- WKS.test(l,C2,alternative="greater")

                                # depleted
pvbl <- WKS.test(b,C2,alternative="less")
pvkl <- WKS.test(k,C2,alternative="less")
pvll <- WKS.test(l,C2,alternative="less")
```

P-values can be visualized using `paired.pvalues`. The entries are two vectors of p-values and a keyword, to be searched among gene set names. Negative logarithms of p-values are plotted. The gene sets whose names match the keyword appear as triangles. They can be identified by clicking on the plot, if `nid` is provided. Title and axes labels are optional.

```
paired.pvalues(pvb,pvl)
paired.pvalues(pvb,pvl,"breast")
paired.pvalues(pvb,pvl,"liver",nid=3)
paired.pvalues(pvbg,pvlg,"liver",nid=3)
paired.pvalues(pvbl,pvll,"liver",nid=3)
{paired.pvalues(pvb,pvl,"liver",
   title="breast vs. liver",xlab="breast",ylab="liver")}
```

False Detection Rate adjustment of p-value vectors is obtained by the R function `p.adjust`.

```
pvba <- p.adjust(pvb,method="BY")
pvka <- p.adjust(pvk,method="BY")
pvla <- p.adjust(pvl,method="BY")
```

Significant p-values can be listed as follows.

```
pvba[pvba<0.05]
pvka[pvka<0.05]
pvla[pvla<0.05]
```

By default, `WKS.test` calculates over ten thousand simulations. This can be modified using the variable `nsim`. The default value for the number of discretization points `ndiscr` is the size of the vector.

```
pvb1 <- WKS.test(b,C2,nsim=1e5,ndiscr=1000)
paired.pvalues(pvb1,pvb)
```

The initial vector can be replaced by its rank statistics. In that case, a more precise calculation is done, using the values of `F1.rds`, which have been pre-calculated over one million simulations.

```
pvbr <- WKS.test(b,C2,ranked=TRUE)
paired.pvalues(pvb,pvbr,"breast")
pvkr <- WKS.test(k,C2,ranked=TRUE)
paired.pvalues(pvk,pvkr,"kidney")
pvlr <- WKS.test(l,C2,ranked=TRUE)
paired.pvalues(pvl,pvlr,"liver")
```

The function `GSEA.test` reproduces the code from [108, 107]. The default value for the number of simulations is one thousand. Even for that reduced number, it is much slower than `WKS.test`. Figure 4.22 in section 4.5 has been produced by the following commands (`GSEA.test` with ten thousand simulations for each gene set takes very long).

```
pvlW <- WKS.test(l,C2,nsim=1e5)
pvlG <- GSEA.test(l,C2,nsim=1e4)
paired.pvalues(pvlG,pvlW,"liver")
```

Two visualizations of enrichment scores are proposed: `cumulated.weights` and `enrichment.plot`. The first one plots the cumulated proportions of weights (function $S_n(t)$ in the paper), which is used in both WKS and GSEA tests. The entries are the vector and the database to be tested, plus a vector of gene set names. All curves are superposed on the same graphic. The second function, `enrichment.plot` plots for each gene set, the difference between the cumulated weights and their expectation under the null hypothesis.

```
               # two-sided, low p-values
gs <- names(pvb)[1:10]
cumulated.weights(b,C2,gs)
enrichment.plot(b,C2,gs)
               # two-sided, high p-values
gs <- names(tail(pvb,10))
cumulated.weights(b,C2,gs)
enrichment.plot(b,C2,gs)
               # right-sided, low p-values = enriched
gs <- names(pvbg)[1:10]
cumulated.weights(b,C2,gs)
enrichment.plot(b,C2,gs)
               # right-sided, high p-values = depleted
gs <- names(tail(pvbg,10))
cumulated.weights(b,C2,gs)
enrichment.plot(b,C2,gs)
               # left-sided, low p-values = depleted
gs <- names(pvbl)[1:10]
cumulated.weights(b,C2,gs)
enrichment.plot(b,C2,gs)
               # left-sided test, high p-values = enriched
gs <- names(tail(pvbl,10))
cumulated.weights(b,C2,gs)
enrichment.plot(b,C2,gs)

cumulated.weights(l,C2,"ACEVEDO_METHYLATED_IN_LIVER_CANCER_DN")
enrichment.plot(l,C2,"ACEVEDO_METHYLATED_IN_LIVER_CANCER_DN")
```

Regarding the DWKS test, there is no R script available online yet. For this reason, all the commands that will be given after, have to be copied and pasted on a terminal. We also consider that the files `C2.rds` and `liver.rds` (see beginning of the section) are available and loaded as before.

The test function provided is: `DWKS.test`. It takes a vector, a database and a distribution function F as its first three entries. Default values are provided for the other variables. The output is a sorted vector of p-values, indexed by the gene set names. The three alternatives are `"greater"`, `"less"`, and `"two.sided"` (default).

In order to estimate the distribution function F from the vector and the database in the way described in subsection 3.2.4, you can use the function `cumdistr`. The entries are the vector and the ("reduced") database plus the number of groups you want to divide the genes into.

```
C2r <- reduce.symbols(names(l),C2)  # reduce the database to the gene symbols
                                    # present in the vector
cdf <- cumdistr(l,C2r,10)
```

Next, the DWKS test can be applied.

```
                                # default: two.sided
pvl <- DWKS.test(l,C2,cdf,alternative="two.sided")

                                # enriched
pvlg <- DWKS.test(l,C2,cdf,alternative="greater")
```

```
                                # depleted
pvll <- DWKS.test(l,C2,cdf,alternative="less")
```

For the visualization of enrichment scores the function `cumulated.weights` can be used. The entries are the vector and the database to be tested, plus a vector of gene set names and the cumulative distribution function F. All curves are superposed on the same graphic.

```
               # two-sided, low p-values
gs <- names(pvl)[1:10]
cumulated.weights(l,C2,gs,cdf)
               # two-sided, high p-values
gs <- names(tail(pvl,10))
cumulated.weights(l,C2,gs,cdf)
               # right-sided, low p-values = enriched
gs <- names(pvlg)[1:10]
cumulated.weights(l,C2,gs,cdf)
               # right-sided, high p-values = depleted
gs <- names(tail(pvlg,10))
cumulated.weights(l,C2,gs,cdf)
               # left-sided, low p-values = depleted
gs <- names(pvll)[1:10]
cumulated.weights(l,C2,gs,cdf)
               # left-sided test, high p-values = enriched
gs <- names(tail(pvll,10))
cumulated.weights(l,C2,gs,cdf)
```

Chapter 5

Conclusion and perspectives

5.1 Conclusion

In the current work, a new method for testing the relative enrichment of a gene set, compared to a vector of numeric data over the whole genome, has been proposed. Like the classical GSEA test of [108], it is based on cumulated proportions of weights, but a different centering is used, making our method more statistically sound. A convergence result that generalizes the classical Kolmogorov Smirnov theorem, has been obtained. The corresponding testing procedure extends the standard Kolmogorov Smirnov test and has been called Weighted Kolmogorov Smirnov (WKS). Some further theoretical properties concerning either the limiting stochastic process or the limiting distribution, not always directly related to the WKS test, are also derived. A major advantage of the WKS test compared to the classical GSEA is that the calculation of p-values only depends on the vector to be tested, and not on the gene set. Therefore, the same distribution function can be used for calculating p-values over many gene sets. This results in very large saving in computing time. A Monte-Carlo evaluation has shown that the procedure is precise for values of the gene set size larger than 40. For a set of less than 40 genes, the WKS test is conservative, in the sense that the p-value is increased, and therefore the gene set is less likely to be declared significant. For statistical coherence, the gene set size should not be larger than 1 000. The WKS test has been compared with the classical GSEA test over expression vectors of tumors coming from the GEO dataset GSE36133 of [10], tested against the MSig database C2 [108]. The comparison has shown that the results of both tests are globally coherent. The WKS test tends to output less significant gene sets out of the whole database, but more out of gene sets specifically related to the same type of tumor. In particular, the WKS test detects sets of underexpressed genes which are not significant for GSEA. Another comparison has been reported, on the GEO dataset GSE3165 of [56]: similar conclusions hold true. Examples where WKS detects clearly enriched gene sets that GSEA fails to detect were given. These encouraging results need to be consolidated, by using the WKS test over different types of vectors, and more databases of gene sets. A comparison between the WKS test and other very known existing methods in terms of power has been conducted over several different parametered families of alternatives. The superiority of one or another test is discussed in each case.

Like the GSEA test, the WKS test can be used on any type of numeric data. In particular, a transformation can be applied to the raw expression levels before testing. In particular, the initial data can be replaced by their ranks, in which case the test has low computing cost, for a good precision. If, over the same database, the p-values of the initial vector, and the vector of ranks are compared, a good agreement is observed; yet less gene sets are declared significant against the rank vector. The WKS was also made one-sided, by testing the signed difference between $S_n(t)$ and $G(t)$: a gene set for which $\inf(S_n(t) - G(t))$ is significantly negative, contains genes whose weights tend to be small (depletion). Conversely, gene sets for which $\sup(S_n(t) - G(t))$ is significantly positive, contain more genes with large weights (enrichment).

Both the GSEA and the WKS tests have been implemented in a R script. It is available online,

together with data samples, and a user manual, from the following address.

`http://ljk.imag.fr/membres/Bernard.Ycart/publis/wks.tgz`

It is also given in section 4.6.3. The WKS test has been implemented in the software tool AutoCompare_ZE-WKS as well, in order to facilitate its use. We hope this will encourage further testing of the tool, and validation in new biological studies.

Then, a generalization of the WKS test that has been called Doubly Weighted Kolmogorov Smirnov (DWKS) test aiming at reducing the large number of false discoveries occuring due to the not so appropriate null hypothesis assumed by both GSEA and WKS has been proposed. It has been shown that the existing model for the gene sets assuming equal probabilities for the genes is very far from what is observed in practice where the gene frequency distribution in all existing databases is clearly right skewed. If the ordering of the genes in the vector of numeric data was independent of the gene frequency, then there would not be any problem: the genes inside the gene set would still be uniformly distributed throughout the list. However, it is shown that in practice the "frequent" genes tend to rank in the first positions of the numeric vector. As a result, a large number of gene sets (containing the frequent genes) has many genes in the first part of the ranked list and ends up being declared significant due to the frequency problem, although having nothing to do with the under examination condition. Even worse, the WKS and GSEA tests tend to output a very large number of significant results in data containing no biological information. For this reason, a distribution function F biased towards the left, is estimated from the data and used-in the place of uniform distribution- to perform the test. The DWKS test addresses efficiently the issue that has led to its construction, by reducing the number of significant results in data with no biological information. More importantly, when applying the DWKS test to the GEO dataset GSE3165, the number of significant gene sets was reduced by half, while the gene sets specifically related to the under examination type of tumor were still output from the test.

Independently from the above contributions, we have also provided a R script that downloads and pretreats expression datasets from a public online repository, the GEO repository in an automated way. In this way, expression datasets are downloaded, annotated and duplicate gene symbols are removed. The expression matrices so obtained are then easy to use to perform any statistical analysis. Our idea is not new, since another package, the inSilicoDb, although very recent, existed already. However, our R script, although much slower than the inSilicoDb package, is more flexible offering the option to choose a method for removing duplicates, easier to use and allows the possibility of downloading datasets that the other package does not.

5.2 Perspectives

- In subsections 3.1.1 and 3.1.3, the one-sample, both two-sided and one-sided (D)WKS test was studied. However, just like the KS test, the one-sample (D)WKS test could be extended to a two-sample (D)WKS test testing if the maximal difference between two cumulated weight functions is significant or not.

- In section 4.5, the (D)WKS test has been applied to real data. Although we consider the results reported there as representative of the observations that can be obtained on different situations, we do not deny that further application of the test could reveal other situations as well where the (D)WKS test outperforms GSEA or inversely. Thus, a more extensive application of the (D)WKS test in practice, could help us gain a better understanding of which gene sets are output as significant.

- In this work, a general testing procedure has been proposed, the (D)WKS test. We showed that one particular application where it could be implemented is when testing the enrichment of a gene set. However, other applications where it could be implemented, can be searched. In particular, the resulting software could find many potential applications, not only for the analysis of microarray profiles, but also for all the other types of genome-scale data that could give rise to a ranked vector of genes associated with weights, like NGS, proteomics or metabolomics data.

In addition, two other applications where the WKS test could replace GSEA are the so-called ssGSEA [9] and the Connectivity Map [73].

- The estimation of the limiting distribution in (D)WKS test is done through Monte-Carlo simulation. Some explicit formulas were derived in subsection 3.1.7 for very special cases. Thus, one could try to derive theoretical (or algorithmic) results for calculation or better approximation of the limiting distribution in other cases too.

- In section 4.1, it was shown that the model of unequal probabilities for the genes is a better model for the database than the one assuming equal probabilities. However, there are still discrepancies between the theoretical model and the actual data. In addition, the problem of false detection rates is not fully solved (see subsection 4.4.2). Thus, better (or other) models for the databases should be derived.

- Regarding small sample sizes, an explicit calculation of the distribution of the test statistic was given for $n = 1$ in subsection 3.1.7. We believe that explicit formulas can also be derived for $n = 2$ or $n = 3$. In addition, in subsection 3.3.1, it has been shown that the WKS test is not very precise and underestimates significance. Thus, the derivation of theoretical (or algorithmic) results concerning the distribution of the (D)WKS test statistic for small sample sizes-better than a Monte-Carlo simulation-could be another direction of research.

- In subsections 3.1.1 and 3.1.3, the (D)WKS test was derived for continuous random variables with sample space the interval $[0, 1]$. The WKS test can be immediately extended to random variables with sample space $\mathbb{R}$. However, it is more tricky to study the problem when the random variables are discrete. In that case, the R package dgof [7] implementing the KS test in the discrete case, could be of help.

Appendix

The functions themselves for data formatting are the following:

```
#-------------------------------------------------------------
#                 Data downloading and formatting
#-------------------------------------------------------------

#------------------ auxiliary functions ------------------------

inst.pack <- function(pkg){
#   Checks if the Bioconductor package pkg is already installed.
#   If not, it installs it.
#
if(!pkg%in%installed.packages()){
 source("http://bioconductor.org/biocLite.R")
 biocLite(pkg,ask=FALSE)
}
}                                       # end function inst.pack

download.data <- function(gsen){
#   Gets the GEO object (GSE) from NCBI corresponding to the integer
#   GSE number gsen and returns it (more specifically a list of
#   ExpressionSet objects).
#
#   Usage: gobject <- download.data(2109)
#
gsename <- {paste("GSE",                # add the prefix 'GSE' to the gsen
           as.character(gsen),sep="")}
inst.pack("GEOquery")
library("GEOquery")
gobject <- getGEO(gsename)              # get the GEO object (GSE) from NCBI
return(gobject)
}                                       # end function download.data

separate.data <- function(eset){
#   Takes a Bioconductors ExpressionSet object eset and separates
#   the three important parts of information it contains: data,
#   supplementary information and information on the annotation.
#   Returns the 3 matrices (two matrices and a data frame) as a list.
#
#   Usage: listdia <- separate.data(gobject[[1]])
#
IM <- t(pData(phenoData(eset)))         # info matrix
dm <- exprs(eset)                       # data matrix
```

```
am <- pData(featureData(eset))          # annotation data frame
return(list(dm=dm,IM=IM,am=am))
}                                       # end function separate.data

omit.na <- function(dm,am){
#   Takes a data matrix dm and a data frame am (same row names) and
#   removes the rows of dm containing NA values. The corresponding
#   rows of am are removed too. The transformed structures are then
#   returned as a list.
#
#   Usage: listda <- omit.na(listdia$dm,listdia$am)
#
if (length(which(is.na(dm)))>0){
 dm <- na.omit(dm)                      # eliminate the rows containing NAs
 am <- am[rownames(dm),]                # eliminate the corresponding rows
}                                       # in the annotation matrix
return(list(dm=dm,am=am))

}                                       # end function omit.na

mapping <- function(x,probe_ids){
#   Takes the object of class "AnnDbBimap" x and the vector
#   of character probe-ids. It returns a vector with the
#   corresponding gene symbols (and the character '' for
#   the probe-ids which do not have a mapping)
#
#   Usage: library(hgu133plus2.db); sym <- hgu133plus2SYMBOL;
#          annotation_vector <- mapping(sym,rownames(listda$dm))
#
mapped_probes <- mappedkeys(x)          # Get the probe identifiers that are mapped
                                        # to a gene symbol
xx <- as.list(x[mapped_probes])         # convert to a list(in this list
                                        # the gene symbols are the elements and the
                                        # names are the corresponding probe-ids)
genesymb <- xx[probe_ids]               # take the gene symbols for the
                                        # probe-ids present in the experiment
                                        # if there was a mapping to null value then replace
genesymb[which(genesymb=='NULL')] <- '' # the NULL with the character ''
names(genesymb) <- NULL

genesymb <- unlist(genesymb)            # convert to a vector
return(genesymb)
}                                       # end function mapping

construct.annotv <- function(platform,listda){
#   Finds for the rownames of matrix listda$dm realized by
#   platform (available set:{"GPL570","GPL96","GPL6947","GPL10558",
#   "GPL15308"}) the corresponding gene symbols.
#   This annotation vector is then returned.
#
#   Usage: annotation_vector <- construct.annotv("GPL570",listda)
#
switch(platform,                        # find annotation vector
```

```
   "GPL570" = {inst.pack("hgu133plus2.db"); library(hgu133plus2.db)
                   sym <- hgu133plus2SYMBOL
                   annotation_vector <- mapping(sym,rownames(listda$dm))},
   "GPL96" = {inst.pack("hgu133a.db"); library(hgu133a.db)
                   sym <- hgu133aSYMBOL
                   annotation_vector <- mapping(sym,rownames(listda$dm))},
   "GPL6947" = {inst.pack("illuminaHumanv3.db"); library(illuminaHumanv3.db)
                   sym <- illuminaHumanv3SYMBOL
                   annotation_vector <- mapping(sym,rownames(listda$dm))},
   "GPL10558" = {inst.pack("illuminaHumanv4.db"); library(illuminaHumanv4.db)
                   sym <- illuminaHumanv4SYMBOL
                   annotation_vector <- mapping(sym,rownames(listda$dm))},
   "GPL15308" = {entrez_ids <- listda$am[,"ORF"]; inst.pack("org.Hs.eg.db")
                   library(org.Hs.eg.db); sym <- org.Hs.egSYMBOL
                   annotation_vector <- mapping(sym, entrez_ids)},
    stop("This experiment cannot be analyzed"))
return(annotation_vector)
}                                       # end function construct.annotv

reduce.data <- function(m1,annotation_vector){
#   Replaces the probe-ids (row names of matrix m1) by the corresponding
#   gene symbols as these are given by the vector of character
#   annotation_vector. It also reduces the data of matrix m1, so that it
#   contains only the information for the probe-ids which have
#   a mapping to a gene symbol (the rows for the probe-ids not
#   having a mapping are removed).
#   Returns the reduced matrix (with row names the gene symbols).
#
#   Usage: m2 <- reduce.data(listda$dm,annotation_vector)
#
genesymb <- annotation_vector           # take the gene symbols for the
                                        # probe-ids
                                        # find the positions for which there
notnullvalues <- which(genesymb!='')    # was a mapping
datavalinitialm <- {matrix(0,           # matrix initialization
   nrow=length(notnullvalues),ncol=ncol(m1))}
datavalinitialm <- m1[notnullvalues,]   # reduced matrix
rownames(datavalinitialm) <- genesymb[notnullvalues] # store the gene symbols as
                                        # rownames
return(datavalinitialm)
}                                       # end function reduce.data

remove.duplicates <- function(m2,g,margin){
#   Removes duplicates in matrix m2. More
#   specifically, in case of duplicates, i.e if
#   a gene is represented in more than one rows
#   then we keep the expression values, as these are
#   defined from the function g and the margin
#   (integer belonging in {1,2}).
#   If margin=1 then we calculate the function g in every row

#   and keep the row (probe) with the maximum calculated value.
#   If margin=2 then we perform the function g in each column (mix probes).
```

```
#   The transformed matrix is returned.
#
#   Usage: remove.duplicates(m2,IQR,1)
#          remove.duplicates(m2,median,2)
#
uniquegenesymb <- unique(rownames(m2))  # collapse gene symbols
lg <- length(uniquegenesymb)            # number of different gene symbols
colnames <- colnames(m2)
colnames(m2) <- NULL
                #-----------------------------------------------------------------
transf <- function(x){
  pos <- {which(rownames(m2)            # numbers of rows containing
          %in%x)}                       # measurements for the same gene
  if (length(pos)>1){                   # if there are more than one,
   if (margin==2){                      # the operation to be performed is defined by margin
    datavalmfin <- apply(m2[pos,],2,g)  # perform the function g in each column
   }else{
    values <- apply(m2[pos,],margin,g)  # calculate the function g for every row
    pmv <- which.max(values)
    datavalmfin <- m2[pos[pmv],]        # keep the row with the maximum calculated value
   }
  }else
                                        # else keep the row as it is
  datavalmfin <- m2[pos,]
  return(datavalmfin)
}
               #------------------------------------------------------------------
lst <- lapply(uniquegenesymb,transf)
datavalmf <- as.data.frame(lst)         # convert to data frame
datavalmf <- t(datavalmf)               # take the transpose
colnames(datavalmf) <- colnames
rownames(datavalmf) <- uniquegenesymb
return(datavalmf)
}                                       # end function remove.duplicates


annot.method <- function(m,annotation_vector,method){
#   Takes a matrix m and returns the matrix after 1)performing
#   the mapping from probe-ids to gene symbols with the help
#   of the vector of character annotation_vector and 2)removing
#   duplicates according to method.
#
#   Usage: annotation(m,annotation_vector,list(IQR,1))
#          annotation(m,annotation_vector,list(max,2))
#          DM <- {annotation(listda$dm,annotation_vector,method)}
#
mred <- reduce.data(m,annotation_vector)# perform annotation
DM <- remove.duplicates(mred,method[[1]],method[[2]])# remove duplicates
return(DM)

}                                       # end function annotation

save.data <- function(DM,IM,gsen){
```

```
#   Saves the data matrix DM in the GSEgsen.rds file and
#   the info matrix IM in the GSEgsen_Info.rds file respectively.
#
#   Usage: IM <- listdia$IM; save(DM,IM,2109)
#
gsename <- {paste("GSE",                # add the prefix 'GSE' to the gsen
           as.character(gsen),sep="")}
filename <- paste(gsename,".rds",sep="")# data file name
saveRDS(DM,file=filename)               # save the matrix with the expression values
filename <- paste(gsename,"_Info",".rds",sep="")  # information file name
saveRDS(IM,file=filename)               # save the matrix with the
                                        # supplementary information
}                                       # end function save.data

#--------------------- main function --------------------------

read.gse <- function(gsen,method=list(IQR,1)){
#   Takes the GSE number gsen and saves two matrices
#   in two separate files. In the first file, the
#   corresponding (to the GSE number) matrix with the expression
#   values is contained (after performing the annotation and
#   removing duplicates according to method (default: maximal
#   interquartile interval)) and in the second, the corresponding
#   matrix with the supplementary information is contained. The file
#   names are GSEgsen.rds and GSEgsen_Info.rds respectively.
#
#   Usage: read.gse(11092)
#          read.gse(7307,method=list(sd,1))
#          read.gse(7307,method=list(max,2))
#
gobject <- download.data(gsen)          # download the GEO object (GSE) from NCBI
listdia <- separate.data(gobject[[1]])  # separate the information
IM <- listdia$IM                        # info matrix

# The treatment for the matrix with the exression values follows.

listda <- omit.na(listdia$dm,listdia$am)# omit NA values
annotation_vector <- {construct.annotv(annotation(gobject[[1]]),
                         listda)}       # find the annotation vector
DM <-  {annot.method(listda$dm,         # perform annotation and
        annotation_vector,method)}      # remove duplicates
save.data(DM,IM,gsen)                   # save info and data matrices
}                                       # end function read.gse

read.data <- function(vgse,method=list(IQR,1)){
#   Takes a vector of GSE number vgse and saves two matrices
#   in two separate files for each number. In fact, the function
#   read.gse is applied repeatedly.
#
#   Usage: vgse <- c(11092,7307); read.data(vgse)
#          read.data(vgse,method=list(max,2))
#
sapply(vgse,function(x){read.gse(x,method)})
```

```
}                                       # end function read.data
```

The functions for WKS testing are:

```
#---------------------- Variables -------------------------------
#   nsim          integer, number of simulations
#   ndiscr        integer, number of discretization points
#   w             named vector of numeric. It contains the
#                 weights, with the corresponding genes being
#                 given as names
#   weights       named vector of numeric, containing the weights
#                 ordered in decreasing order or a continuous function of it
#   gene.list     vector of character, containing the genes
#                 (names of weights)
#   db            list of vector of character, database of gene sets
#   gene.set      vector of character, gene set
#   gene.sets     vector of character, gene set names
#   ranked        logical, indicates whether the true values in w should
#                 be replaced by their ranks or not
#   alternative   character in "two.sided", "greater", "less". Specifies
#                 the alternative hypothesis of the WKS test
#   s             vector of numeric, a sample
#   pv1,pv2       named vectors of numeric, p-values
#   kwd           character. A keyword, usually a tissue
#                 (e.g. "liver","breast","lung", etc.)
#   nid           integer, number of points to be identified on a graph
#   v1,v2,pv      vectors of character
#   lens          list of numeric, containing the gene set sizes
#   title         character, title of a graphic
#   xlab          character, label for the x-axis of a graphic
#   ylab          character, label for the y-axis of a graphic

#---------------------- Functions -------------------------------

#------------------- Main functions-------------------------------
                    # Statistical tests
# WKS.test(w,db,nsim=10000,ndiscr,ranked=FALSE,alternative="two.sided")
# GSEA.test(w,db,nsim=1000)
                    # Graphical comparison of two p-value vectors
# paired.pvalues(pv1,pv2,kwd,nid=0,title="Paired p-values",
#                  xlab="-log10 p-values",
#                  ylab="-log10 p-values")
                    # Graphical plot of gene sets
# cumulated.weights(w,db,gene.sets,title="Cumulated weights")
# enrichment.plot(w,db,gene.sets,xlab="genes",ylab="ES")

#----------------- Auxiliary functions-------------------------
                    # Calculation of the test statistics
# WKS.EnrichScore(gene.list, gene.set, weights,alternative)
# GSEA.EnrichScore(gene.list, gene.set, weights)
                    # Simulation for the limiting stochastic process
# lim.distr(weights,nsim,ndiscr,alternative)
# rbrmotint(weights,ndiscr)
```

```
                    # comparison of gene sets
# order.matching(v1,v2)
#--------------- global graphic parameters ---------------------
colpoint <- "blue3"                  # color for points
coltri <- "red3"                     # color for triangles
colline <- "black"                   # color for lines
colstep <- "blue3"                   # color for step function
colid <- "green4"                    # color for identifying
cexp <- 0.2                          # point size
lth <- 1.5                           # line thickness
lty <- 2                             # line type
cexa <- 1.1                          # axes size
cexl <- 1.1                          # labels size
cexm <- 1.3                          # title size
cext <- 0.7                          # text size
sctick <- 0.1                        # scale for ticks

#-----------------------------------------------------------------
#                     Statistical tests
#-----------------------------------------------------------------


#-----------------------------------------------------------------
#                         WKS test
#-----------------------------------------------------------------

WKS.test <- function(w,db,nsim=10000,ndiscr,ranked=FALSE,alternative="two.sided"){
#   Takes a named vector w and a database db. Performs
#   the WKS test with alternative "two.sided", "less", or "greater".
#   The limiting distribution is estimated through nsim Monte-Carlo
#   simulations, the number of discretization points being ndiscr.
#   If ndiscr is missing, then the number of discretization points
#   is equal to the size of w.
#   Calculates the p-values (upper bound of the one-sided confidence
#   interval with level 0.95), sorts them and returns them as a vector.
#   If ranked=TRUE, the weights are  replaced by their ranks, and the
#   pre-calculated F1.rds and F1os.rds values for the two sided and
#   one sided WKS test respectively, are used.
#
#   Usage: l <- readRDS("liver.rds")
#          C2 <- readRDS("C2.rds")
#          pvwks <- WKS.test(l,C2);
#          pvwks <- WKS.test(l,C2,alternative="greater");
#          pvwks <- WKS.test(l,C2,ranked=TRUE)
#          pvwks <- WKS.test(l,C2,ranked=TRUE,alternative="less")
#
correl.vector <- sort(w,decreasing=TRUE)# sort the vector in decreasing order
lg <- length(correl.vector)
gene.list <- names(correl.vector)       # all genes
if (ranked){
 if (alternative=="two.sided")
   ld <- readRDS("F1.rds") else
   ld <- readRDS("F1os.rds")
```

```
correl.vector <- rank(correl.vector)/lg # replace true values by ranks
ots <- {unlist(lapply(db,function(x){   # observed test statistics
       WKS.EnrichScore(gene.list, x, correl.vector,alternative)}))}
ordots <- ots[order(ots,decreasing=TRUE)]  # ordered in decreasing order
ordots <- na.omit(ordots)               # eliminate NA values
                                        # (corresponding to gene sets with
                                        # an empty intersection with the list)
pvalsm <- {unlist(lapply(ordots,function(x){
       length(which(ld>=x))}))}
                                        # upper bound of the one-sided confidence
pvalsci <- unlist(lapply(pvalsm,function(x) # interval with level 0.95
           {prop.test(x,1000000,alternative="less")$conf.int[2]}))
}else{
ots <- {unlist(lapply(db,function(x){   # observed test statistics
       WKS.EnrichScore(gene.list, x, abs(correl.vector),alternative)}))}
ordots <- ots[order(ots,decreasing=TRUE)]  # ordered in decreasing order
ordots <- na.omit(ordots)               # eliminate NA values
                                        # (corresponding to gene sets with
                                        # an empty intersection with the list)
if (missing(ndiscr)){
ld <- {lim.distr(weights=abs(correl.vector),
       nsim=nsim,ndiscr=lg,alternative)} # estimation of limiting distribution
}else{
ld <- {lim.distr(weights=abs(correl.vector),
       nsim=nsim,ndiscr=ndiscr,alternative)}
}
pvalsm <- {unlist(lapply(ordots,function(x){
       length(which(ld>=x))}))}
                                        # upper bound one-sided confidence
pvalsci <- unlist(lapply(pvalsm,function(x) # interval with level 0.95
           {prop.test(x,nsim,alternative="less")$conf.int[2]}))
}

return(pvalsci)
}                                       # end function WKS.test


#-----------------------------------------------------------------
#                        GSEA test
#-----------------------------------------------------------------


GSEA.test <- function(w,db,nsim=1000){
#   Takes a named vector w and a database db. Performs the GSEA test.
#   The number of Monte-Carlo simulations for every gene set
#   is nsim. Calculates the p-values (upper bound of the one-sided confidence
#   interval with level 0.95), sorts them and returns them as a vector.
#
#   Usage: dcol <- readRDS("liver.rds")
#          C2 <- readRDS("C2.rds")
#          pvgsea <- GSEA.test(dcol,C2)
#
weights <- sort(w,decreasing=TRUE) # sort weights in decreasing order
```

```
gene.list <- names(weights)        # gene names
db <- reduce.symbols(gene.list,db) # reduce database to symbols in gene.list
Ng <- length(db)                   # number of gene sets
Obs.ES <- vector(length=Ng, mode="numeric")
lg <- vector(length=Ng, mode="numeric")
lens <- lapply(db,length)
for (i in 1:Ng) {
  Obs.ES[i] <- {GSEA.EnrichScore(gene.list,
                       db[[i]],weights)}
}
phi <- matrix(nrow = Ng, ncol = nsim)
for (r in 1:nsim) {
   rdb <- random.database(lens,gene.list)# take random gene sets
   for (i in 1:Ng) {
                                         # calculate test statistic
    phi[i, r] <- {GSEA.EnrichScore(gene.list,
                             rdb[[i]],weights)}
   }
}

p.vals <- matrix(0, nrow = Ng, ncol = 1)

for (i in 1:Ng) {
  ES.value <- Obs.ES[i]
  if (ES.value >= 0) {                    # estimate significance according to the
                                          # sign of the observed test statistic
  temp <- phi[i,which(phi[i,]>=0)]
  p.vals[i, 1] <- {signif(length(which(temp >= ES.value))/
                   length(temp), digits=5)}
  } else {
  temp <- phi[i,which(phi[i,]<0)]
  p.vals[i, 1] <- {signif(length(which(temp <= ES.value))/
                   length(temp), digits=5)}
  }
}
indna <- which(is.na(p.vals))             # replace the na p-values
p.vals[indna] <- 0                        # with zero

                                          # upper bound of the one-sided confidence
p.vals <- unlist(lapply(p.vals*nsim,function(x) # interval with level 0.95
           {prop.test(x,nsim,alternative="less")$conf.int[2]}))
names(p.vals) <- names(db)
p.valss <- sort(p.vals)                   # sort p-values
return(p.valss)
}                                         # end function GSEA.test

random.database <- function(lens,gene.list){
#   Returns random vectors of sizes given in list
#   lens out of gene.list.
#
return(lapply(lens,function(n){sample(gene.list,n)}))
}                                        # end function random.database
```

```
reduce.symbols <- function(pv,db){
#   Reduces database db to symbols present in vector pv.
#   Returns the reduced database.
#
rdb <- {lapply(db,function(v){          # apply to all gene sets
     return(find.matching(v,pv))})}     # reduce gene set to symbols in pv
l <- lapply(rdb,length)                 # new gene set lengths
rdb <- rdb[which(l>0)]                  # remove empty gene sets
return(rdb)
}                                       # end function reduce.symbols

find.matching <- function(v1,v2){
#    returns the character chains common to
#    v1 and v2, any two vectors of character chains
#
return(v2[which(v2 %in% v1)])
}                                      # end function find.matching

order.matching <- function(v1,v2){
#    returns the indices of those entries of v1 found in v2,
#    v1 and v2 being any two vectors of charater chains
#
return(which(v1 %in% v2))
}                                      # end function order.matching

#-----------------------------------------------------------------
#             Calculation of the test statistics
#-----------------------------------------------------------------

#------------------ auxiliary functions --------------------------

normal.function <- function(weights){
#   Takes a vector weights which contains the discretized
#   values of a function g. Approximates the integral and
#   normalizes it so that it sums up to 1.
#
distr1 <- cumsum(weights)               # calculate (approximate) the integral and
distr <- distr1/distr1[length(weights)] # normalize it so that it sums up to 1
return(distr)
}                                       # end function normal.function


calc_max_diff <- function(weights,s,alternative){
#   Takes a sample s and a vector weights.
#   Calculates and returns the value of the test
#   statistic used for the test with alternative
#   hypothesis alternative.
#
ss <- sort(s)                           # sorted sample
lg <- length(s)                         # sample size
wss <- weights[ss]                      # corresponding weights
```

```
rts <- cumsum(wss)                      # random value of the test statistic
rtsn <- rts/rts[lg]                     # same normalized
distr <- normal.function(weights)       # normalized integral
trval <- distr[ss]                      # expected weighted distribution
switch(alternative,
  "two.sided"={diff1 <- rtsn-trval      # difference at the discontinuity points
     if (lg>=2){
        diff2 <- {c(trval[1],trval[2:lg]-rtsn[1:(lg-1)])}
                                        # and at one point before
     } else{
        diff2 <- trval[1]
     }
     diff <- abs(c(diff1,diff2))},
  "greater"={diff <- rtsn-trval},
  "less"={
     if (lg>=2){
        diff <- {c(trval[1],trval[2:lg]-rtsn[1:(lg-1)])}
                                        # at one point before
     } else{
        diff <- trval[1]
     }})
m <- max(diff)                          # take the maximum
return(m)
}                                       # end function calc_max_diff


WKS.EnrichScore <- function(gene.list, gene.set, weights,alternative){
#   Calculates the value of the WKS test statistic
#   when the alternative hypothesis is alternative for
#   given numeric data weights, indexed by the
#   genes in gene.list and a given set of genes gene.set.
#   Returns the calculated observed test statistic.
#   Usage: dcol <- readRDS("kidney.rds")
#          C2 <- readRDS("C2.rds")
#          weights <- sort(dcol,decreasing=TRUE)
#          gene.list <- names(weights)
#          gene.set <- C2[[16]]
#          WKS.EnrichScore(gene.list, gene.set, abs(weights),
#                          alternative="two.sided")
#          WKS.EnrichScore(gene.list, gene.set, abs(weights),
#                          alternative="greater")
#
cgi <- order.matching(gene.list,gene.set)# common gene indices
if (length(cgi)>0)
m <- {calc_max_diff(
     weights=weights,s=cgi,
     alternative)}                      # calculate test statistic
else m <- NA
return(m*sqrt(length(cgi)))             # scale it
}                                       # end function WKS.EnrichScore


GSEA.EnrichScore <- function(gene.list, gene.set, weights) {
```

```
#   Calculates the value of the GSEA test statistic for
#   given numeric data weights, ranked in decreasing order, indexed by the
#   genes in gene.list and a given set of genes gene.set.
#   Returns the calculated observed test statistic.
#   Part of the functions encoded by the Broad Institute, have been used.
#   Usage: dcol <- readRDS("kidney.rds")
#          C2 <- readRDS("C2.rds")
#          weights <- sort(dcol,decreasing=TRUE)
#          gene.list <- names(weights)
#          gene.set <- C2[[16]]
#          GSEA.EnrichScore(gene.list, gene.set, weights)
#
                                        # get signs
   tag.indicator <- sign(match(gene.list, gene.set, nomatch=0))
   no.tag.indicator <- 1 - tag.indicator
   N <- length(gene.list)               # length of gene list
   Nh <- length(gene.set)               # gene set length
   Nm <-  N - Nh                        # difference between the two
   correl.vector <- abs(weights)
   sum.correl.tag    <- sum(correl.vector[tag.indicator == 1])
   norm.tag    <- 1.0/sum.correl.tag     # normalization factor
   norm.no.tag <- 1.0/Nm                # 1/(N-Nh)
                                        # calculation of the test statistic
   RES <- {cumsum(tag.indicator*correl.vector * norm.tag
                  - no.tag.indicator * norm.no.tag)}
   max.ES <- max(RES)
   min.ES <- min(RES)
   if (max.ES > - min.ES) {
      ES <- signif(max.ES, digits = 5)
   } else {
      ES <- signif(min.ES, digits=5)
   }
   return(ES)
}                                       # end function GSEA.EnrichScore


#-------------------------------------------------------------------
#        Simulation for the limiting stochastic process
#-------------------------------------------------------------------

#------------------ auxiliary function --------------------------

rbrmotint <- function(weights,ndiscr){
#   Takes a vector weights.
#   Simulates a trajectory of the stochastic integral
#   of weights with respect to BM in ndiscr
#   discretization points. Returns the result.
#
incr <- rnorm(n=ndiscr,sd=1/sqrt(ndiscr))# BM increments
lg <- length(weights)
subseq <- trunc(seq(1,lg,lg/ndiscr))
mult <- incr*weights[subseq]            # multiply increments
res <- cumsum(mult)                     # sum of increments
```

```
return(res)
}                                       # end function rbrmotint

#--------------------- main function ----------------------------

lim.distr<-function(weights,nsim,ndiscr,alternative){
#   Takes a vector weights. Simulates nsim trajectories
#   of the limiting stochastic process (centered BM integral).
#   The number of discretization points for each is ndiscr.
#   Returns a vector with the realizations of the limiting
#   random variable (maximum of the absolute simulated trajectory
#   when alternative="two.sided" or maximum of the simulated
#   trajectory else).
#   Usage: dcol <- readRDS("kidney.rds")
#          weights <- sort(dcol,decreasing=TRUE)
#          lg <- length(weights)
#          ld <- lim.distr(abs(weights),1000,lg,alternative="two.sided")
#          plot(ecdf(ld))
#          ld2 <- lim.distr(abs(weights),1000,lg,alternative="greater")
#          lines(ecdf(ld2),col=2)
#
lg <- length(weights)
subseq <- trunc(seq(1,lg,lg/ndiscr))
distr1 <- cumsum(weights)               # approximate the integral and
distr <- distr1/distr1[lg]              # normalize it so that it sums up to 1
norm_fact <- distr1[lg ]/lg             # normalization factor

fres <- vector(nsim,mode='numeric')     # vector initialization
for(k in 1:nsim){
 res <- rbrmotint(weights,ndiscr)       # simulation of the BM integral
                                        # realization of process
 resnorm <- res-distr[subseq]*res[ndiscr]
 if (alternative=="two.sided"){
   fres[k] <- max(abs(resnorm))         # maximum of the absolute limit
 } else {
   fres[k] <- max(resnorm)              # asymptotic random variable
 }
}                                       # end for
return(fres/norm_fact)                  # scale the result
}                                       # end function lim.distr



#-----------------------------------------------------------------
#                Plot two vectors of p-values
#-----------------------------------------------------------------

paired.pvalues <- function(pv1,pv2,kwd,nid=0,title="paired p-values",
                  xlab="-log10 p-values",ylab="-log10 p-values"){
#   Takes two vectors of p-values pv1 and pv2.
#   Plots the two vectors (pv1 on the y-axis, pv2 on the x-axis),
#   -log10 transformed, and represents the gene sets
#   matching kwd as red triangles. Identifies nid of those.
```

```
#
#   Usage: paired.pvalues(pvwks,pvgsea,"liver",title="WKS vs. GSEA",
#            xlab="-log10 p-values GSEA",ylab="-log10 p-values WKS")
#
lpv1 <- -log10(pv1)                     # -log10 transformed
lpv2 <- -log10(pv2[names(pv1)])         # -log10 transformed
{plot(lpv1,lpv2,                        # all points
     xlab=xlab,ylab=ylab,
     main=title,cex.axis=cexa,cex.lab=cexl,
     cex.main=cexm,pch=19,cex=cexp,col="blue4")}
abline(0,1,lty=lty,lwd=lth)             # add bisector
abline(h=-log10(0.05),lty=lty,lwd=lth)  # horizontal line at pv=0.05
abline(v=-log10(0.05),lty=lty,lwd=lth)  # vertical line at pv=0.05
if (!missing(kwd)){
   lkwd <- tolower(kwd)                 # switch to lower case
   nmpv <- tolower(names(pv1))          # switch to lower case
   indt <- which(grepl(lkwd,nmpv))      # get gene sets matching keyword
   lpv1t <- lpv1[indt]                  # their p-values for test 1
   lpv2t <- lpv2[indt]                  # their p-values for test 2
   {points(lpv1t,lpv2t,                 # points with biological info
      pch=17,cex=1.2,col=coltri)}       # as red triangles
   if(nid>0){
   {identify(lpv1t,lpv2t,n=nid,         # identify nid gene sets
   labels=names(lpv1t),col=colid,cex=cext)}}
}
}                                       # end function paired.pvalues

#-----------------------------------------------------------------
#               Plot cumulated proportions of weights
#-----------------------------------------------------------------

cumulated.weights <- function(w,db,gene.sets,title="Cumulated weights"){
#   Takes a named vector w, a database db and a vector of
#   gene set names gene.sets.
#   Plots the realizations of the random quantity used in
#   WKS test statistic for the given gene sets.
#   The bisector and the expected theoretical function are
#   added on the graph.
#
#   Usage: dcol <- readRDS("kidney.rds")
#          C2 <- readRDS("C2.rds");
#          gene.sets <- c("ACEVEDO_LIVER_CANCER_DN","ACEVEDO_LIVER_CANCER_UP")
#          cumulated.weights(dcol,C2,gene.sets)
#
xlab <- "t"
ylab <- "Sn(t)"
weights <- abs(sort(w,decreasing=TRUE))        # sort weights in decreasing order
gene.list <- names(weights)                    # all genes
lgw <- length(weights)                         # their number
stpfunc <- function(gene.set){
   ma <- order.matching(gene.list,gene.set)    # indices of common genes
   ma1 <- ma/lgw                               # proportion
   vma <- weights[ma]                          # corresponding weights
```

```
   fvma <- cumsum(vma)                          # cumulated weights
   lg <- length(fvma)
   fvman <- fvma/fvma[lg]                       # normalize
   sfun <- stepfun(ma1,c(0,fvman), f = 0)       # make it a step function
return(sfun)
}                                               # end step function
sfun <- stpfunc(db[[gene.sets[1]]])             # compute first step function
{plot(sfun,verticals = TRUE,do.points=FALSE,    # plot it
     xlim=c(0,1),ylim=c(0,1),col=colstep,
     xlab=xlab,ylab=ylab,
     main=title,cex.axis=cexa,cex.main=cexm)}
len <- length(gene.sets)                        # get number of plots
if (len>1){                                     # if more than one
 {lapply(gene.sets[2:len],                      # iterate
  function(x){lines(stpfunc(db[[x]]),xlim=c(0,1),
  verticals = TRUE,do.points=FALSE,col=colstep)})}
}
abline(0,1,lty=lty,lwd=lth,col=colline)         # add bisector
distr1 <- cumsum(abs(weights))                  # calculate the integral and
distr <- distr1/distr1[lgw]                     # normalize
{lines(seq(1/lgw,1,1/lgw),distr,
   lty=lty,lwd=lth,col=colline)}                # add the expected theoretical
}                                               # end function cumulated.weights


enrichment.plot <- function(w,db,gene.sets,xlab="genes",ylab="ES"){
#   Takes a named vector w, a database db and a vector of
#   gene set names gene.sets and creates a matrix of
#   ceiling(length(gene.sets)/2) x 2 plots.
#   For each gene set, the difference between the cumulated
#   weights (S_n) and the primitive function (G) is plotted.
#   At the bottom of the graph, vertical lines are added,
#   indicating the positions of the genes inside the gene set
#   in the ranked list.
#
#   Usage: dcol <- readRDS("kidney.rds")
#          C2 <- readRDS("C2.rds");
#          gene.sets <- c("ACEVEDO_LIVER_CANCER_DN","ACEVEDO_LIVER_CANCER_UP")
#          enrichment.plot(dcol,C2,gene.sets)
#          pvwks <- WKS.test(dcol,C2)
#          gene.sets <- names(pvwks)[1:5]; enrichment.plot(dcol,C2,gene.sets)
#          gene.sets <- names(tail(pvwks,5)); enrichment.plot(dcol,C2,gene.sets)
#
len <- length(gene.sets)                        # number of gene sets
weights <- abs(sort(w,decreasing=TRUE))         # sort weights in decreasing order
gene.list <- names(weights)                     # all genes
lgw <- length(weights)                          # their number
distr1 <- cumsum(abs(weights))                  # calculate the integral and
distr <- distr1/tail(distr1,1)                  # normalize
plotenrich <- function(gene.set,cexmain){
   name <- gene.set
   gene.set <- db[[gene.set]]
   ma <- order.matching(gene.list,gene.set)     # indices of common genes
```

```
   ma1 <- ma/lgw                                # proportion
   vma <- weights[ma]                           # corresponding weights
   fvma <- cumsum(vma)                          # cumulated weights
   lg <- length(fvma)
   fvman <- fvma/tail(fvma,1)                   # normalize
   trv <- distr[ma]                             # (expected) theoretical values
   X <- rbind(ma1,ma1)
   X <-  as.vector(X)
   X <- c(0,X,1)                                # ordinates
   Y <- rbind(fvman,fvman)
   Y <- as.vector(Y)
   Y <- c(0,0,Y)                                # coordinates for the step function
   C <-  rbind(trv,trv)
   C <- as.vector(C)
   C <- c(0,C,1)
   Y <- Y-C                                     # coordinates for the difference
   scy0 <- min(Y); scy1 <- scy0+(max(Y)-scy0)*sctick
   scyM <- max(Y)
                                                # plot difference S_n-G
   {plot(X,Y,xlim=c(0,1),ylim=c(2*scy0-scy1,scyM),
      type="l",lty=1,col=colstep,
      xlab=xlab,ylab=ylab,main=name,
      cex.lab=cexl,cex.main=cexmain,cex.axis=cexa)}
   lines(c(0,1),c(0,0),lty=lty,col=colline)    # add the line y=0
   scy0 <- min(Y); scy1 <- scy0+(max(Y)-scy0)*sctick
   X <- rbind(ma1,ma1)
   Y <- rbind(rep(2*scy0-scy1,lg),rep(scy0,lg))
   matlines(X,Y,lty=rep(1,lg),col=colline)     # add the vertical lines
}
                                                # split graphical window
if(len>1){
   par(mfrow=c(ceiling(len/2),2))               # more than one gene set
   par(mar = c(4,4,1.5,1.5))                    # margin parameters
                                                # apply to all gene sets
   pl <- lapply(gene.sets,function(x){plotenrich(x,cext)})
      }else{                                    # only one gene set
   plotenrich(gene.sets,cexm)
   }                                            # end if
}                                               # end function enrichment.plot
```

The functions that have been encoded for the DWKS test are the following:

```
#--------------- global graphic parameters ---------------------
colpoint <- "blue3"                  # color for points
coltri <- "red3"                     # color for triangles
colline <- "black"                   # color for lines
colstep <- "blue3"                   # color for step function
colid <- "green4"                    # color for identifying
cexp <- 0.2                          # point size
cext <- 1.5                          # triangle size
lth <- 1.5                           # line thickness
lty <- 2                             # line type
cexa <- 1.1                          # axes size
```

```
cexl <- 1.1                          # labels size
cexm <- 1.3                          # title size
cext <- 0.7                          # text size
#-------------------------------------------------------------
#                     DWKS test
#-------------------------------------------------------------

DWKS.test <- function(w,db,distr,nsim=10000,ndiscr,ranked=FALSE,alternative="two sided"){
#   Takes a named vector w and a database db. Performs
#   the WWKS test (testing if the indices of common genes
#   follow distribution distr).
#   The limiting distribution is estimated
#   through nsim Monte-Carlo simulations with the number
#   of discretization points being ndiscr. (if the argument
#   ndiscr is missing then the number of discretization points
#   is equal to the number of numeric data)
#   Calculates the p-values (upper bound of the one-sided
#   confidence interval with level 0.95), sorts them
#   and returns them as a vector.
#
#   Usage: l <- readRDS("liver.rds")
#          C2 <- readRDS("C2.rds")
#          sq <- seq(1/length(l),1,1/length(l)); distr <- pbeta(sq,shape1=0.5,shape2=0.5)
#          pvwks <- DWKS.test(l,C2,distr=distr)
#          pvwks <- DWKS.test(l,C2,distr=distr,alternative="greater")
#
correl.vector <- sort(w,decreasing=TRUE)# sort the vector in decreasing order
lg <- length(correl.vector)             # length of it
gene.list <- names(correl.vector)       # all genes
if (ranked){
 if (alternative=="two.sided")
   ld <- readRDS("F1.rds") else
   ld <- readRDS("F1os.rds")
correl.vector <- 1-distr                # replace true values by ranks
ots <- {unlist(lapply(db,function(x){   # observed test statistics
       DWKS.EnrichScore(gene.list, x, correl.vector,distr,alternative)}))}
ordots <- ots[order(ots,decreasing=T)]  # ordered in decreasing order
pvalsm <- {unlist(lapply(ordots,function(x){
       length(which(ld>=x))}))}
                                        # upper bound of the one-sided confidence
pvalsci <- unlist(lapply(pvalsm,function(x) # interval with level 0.95
           {prop.test(x,1000000,alternative="less")$conf.int[2]}))
}else{
ots <- {unlist(lapply(db,function(x){   # observed test statistics
       DWKS.EnrichScore(gene.list, x, abs(correl.vector),distr,alternative)}))}
ordots <- ots[order(ots,decreasing=T)]  # ordered in decreasing order
if (missing(ndiscr)){
ld <- {lim_distr(weights=abs(correl.vector),
       nsim=nsim,ndiscr=lg,distr=distr,alternative)}# estimation of limiting distribution
}else{
ld <- {lim_distr(weights=abs(correl.vector),
       nsim=nsim,ndiscr=ndiscr,distr=distr,alternative)}
}
```

```
pvalsm <- {unlist(lapply(ordots,function(x){
       length(which(ld>=x))}))}
                                        # upper bound of the one-sided confidence
pvalsci <- unlist(lapply(pvalsm,function(x) # interval with level 0.95
           {prop.test(x,nsim,alternative="less")$conf.int[2]}))
}
return(pvalsci)
}                                       # end function DWKS.test

#-----------------------------------------------------------------
#        Simulation for the limiting stochastic process
#-----------------------------------------------------------------

#------------------ auxiliary function -------------------------
rbrmotint <- function(weights,ndiscr,distr){
#   Takes a vector weights.
#   Simulates a trajectory of the stochastic integral
#   of weights with respect to BM with time change
#   distr in ndiscr discretization points
#   (\int_0^t g(u)dW(F(u))). Returns the result.
#

                         #-----------------------------------------------------------------
incr <- rnorm(n=ndiscr)                 # i.i.d. N(0,1)
lg <- length(weights)
subseq <- trunc(seq(1,lg,lg/ndiscr))
lg1 <- length(distr)
distrsh <- c(0,distr[1:(lg1-1)])
diff <- distr-distrsh
mult <- incr*weights[subseq]*sqrt(diff) # multiplication with g
                                        # (g(t_i)(W(F_{t_i})-W(F_{t_{i-1}})))
res <- cumsum(mult)                     # calculation of the sum
                                        # sum(g(t_i)(W(F_{t_i})-W(F_{t_{i-1}})))
return(res)
}                                       # end function rbrmotint

#--------------------- main function -----------------------------

lim_distr<-function(weights,nsim,ndiscr,distr,alternative){
#   Takes a vector weights and a distribution distr.
#   Simulates nsim trajectories
#   of the limiting stochastic process (centered BM integral).
#   The number of discretization points for each is ndiscr.
#   Returns a vector with the realizations of the limiting
#   random variable (maximum of the absolute simulated trajectory).
#   Usage: dcol <- readRDS("kidney.rds")
#          weights <- sort(dcol,decreasing=TRUE)
#          lg <- length(weights)
#          sq <- seq(1/length(l),1,1/length(l)); distr <- pbeta(sq,shape1=0.5,shape2=0.5)
#          ld <- lim_distr(abs(weights),1000,lg,distr)
#          plot(ecdf(ld))
#
```

```
lg <- length(weights)
subseq <- trunc(seq(1,lg,lg/ndiscr))
lg1 <- length(distr)
distrsh <- c(0,distr[1:(lg1-1)])        # F_{t_{i-1}}
diff <- distr-distrsh                   # differences F_{t_i}-F_{t_{i-1}}
intg <- cumsum(weights*diff)            # calculate (approximate) the integral
                                        # (\int_0^t g(u)dF)
norm_fact <- intg[lg]                   # normalization factor (\int_0^1 g(u)dF)
intgn <- intg/norm_fact                 # normalize it so that it sums up to 1

fres <- vector(nsim,mode='numeric')     # vector initialization
for(k in 1:nsim){
 res <- rbrmotint(weights,ndiscr,distr) # simulation of the time-changed BM integral
 resnorm <- res-intgn[subseq]*res[ndiscr] # realization of the limiting stochastic process
 if (alternative=="two.sided"){
   fres[k] <- max(abs(resnorm))         # maximum of the absolute limiting stochastic process
 } else {
   fres[k] <- max(resnorm)              # realization of the limiting random variable
 }
}                                       # end for
return(fres/norm_fact)                  # scale the result
}                                       # end function lim_distr



#-----------------------------------------------------------------
#             Calculation of the test statistics
#-----------------------------------------------------------------

#------------------ auxiliary function --------------------------
calc_max_diff <- function(weights,s,distr,alternative){
#   Takes a sample s, a vector weights and a vector
#   with the null distribution distr.
#   Calculates and returns the value of the test
#   statistic.
#
ss <- sort(s)                           # sorted sample
lg <- length(s)                         # sample size
wss <- weights[ss]                      # corresponding weights
rts <- cumsum(wss)                      # random quantity of the test statistic
rtsn <- rts/rts[length(rts)]            # random quantity of the test statistic normalized
lg1 <- length(distr)
distrsh <- c(0,distr[1:(lg1-1)])        # F_{t_{i-1}}
diff <- distr-distrsh                   # differences F_{t_i}-F_{t_{i-1}}
intg <- cumsum(weights*diff)            # calculate (approximate) the integral
norm_fact <- intg[lg1]                  # normalization factor
intgn <- intg/norm_fact                 # normalize it so that it sums up to 1
trval <- intgn[ss]                      # expected weighted distribution

switch(alternative,
  "two.sided"={diff1 <- rtsn-trval      # difference at the discontinuity points
     if (lg>=2){
        diff2 <- {c(trval[1],trval[2:lg]-rtsn[1:(lg-1)])}
```

```
                                        # and at one point before
     } else{
        diff2 <- trval[1]
     }
     diff <- abs(c(diff1,diff2))},
  "greater"={diff <- rtsn-trval},
  "less"={
     if (lg>=2){
        diff <- {c(trval[1],trval[2:lg]-rtsn[1:(lg-1)])}
                                        # and at one point before
     } else{
        diff <- trval[1]
     }})

m <- max(diff)                          # take the maximum
return(m)
}                                       # end function calc_max_diff


find.matching <- function(v1,v2){
#    returns the character chains common to
#    v1 and v2, any two vectors of character chains
#
return(v2[which(v2 %in% v1)])
}                                       # end function find.matching

order.matching <- function(v1,v2){
#    returns the indices of those entries of v1 found in v2,
#    v1 and v2 being any two vectors of charater chains
#
return(which(v1 %in% v2))
}                                       # end function order.matching

#--------------------- main function ----------------------------

DWKS.EnrichScore <- function(gene.list,gene.set,weights,distr,alternative){
#   Calculates the value of the WWKS test statistic for
#   given numeric data weights, indexed by the
#   genes in gene.list and a given set of genes gene.set.
#   The indices of common genes are supposed to follow
#   distribution distr under the null hypothesis.
#   Returns the calculated observed test statistic.
#   Usage: dcol <- readRDS("kidney.rds")
#          C2 <- readRDS("C2.rds")
#          weights <- sort(dcol,decreasing=TRUE)
#          gene.list <- names(weights)
#          gene.set <- C2[[16]]
#          sq <- seq(1/length(l),1,1/length(l)); distr <- pbeta(sq,shape1=0.5,shape2=0.5)
#          WWKS.EnrichScore(gene.list,gene.set,abs(weights),distr)
#
cgi <- order.matching(gene.list,gene.set)# common gene indices
m <- {calc_max_diff(
     weights=weights,s=cgi,distr,alternative)}       # calculate test statistic
```

```
return(m*sqrt(length(cgi)))              # scale it
}                                        # end function WWKS.EnrichScore


#-----------------------------------------------------------------
#               Plot cumulated proportions of weights
#-----------------------------------------------------------------

cumulated.weights <- function(w,db,gene.sets,distr,title="Cumulated weights"){
#   Takes a named vector w, a database db and a vector of
#   gene set names gene.sets.
#   Plots the realizations of the random quantity used in
#   WWKS test statistic for the given gene sets.
#   The indices of common genes are supposed to follow
#   distribution distr according to the null hypothesis.
#   The bisector and the expected theoretical function are
#   added on the graph.
#   Usage: dcol <- readRDS("kidney.rds")
#          C2 <- readRDS("C2.rds");
#          gene.sets <- c("ACEVEDO_LIVER_CANCER_DN","ACEVEDO_LIVER_CANCER_UP")
#          sq <- seq(1/length(l),1,1/length(l)); distr <- pbeta(sq,shape1=0.5,shape2=0.5)
#          cumulated.weights(dcol,C2,gene.sets,distr)
#
xlab <- "t"
ylab <- "Sn(t)"
weights <- abs(sort(w,decreasing=TRUE))        # sort weights in decreasing order
gene.list <- names(weights)                    # all genes
lgw <- length(weights)                         # their number
stpfunc <- function(gene.set){
   ma <- order.matching(gene.list,gene.set)    # indices of common genes
   ma1 <- ma/lgw                               # proportion
   vma <- weights[ma]                          # corresponding weights
   fvma <- cumsum(vma)                         # cumulated weights
   lg <- length(fvma)
   fvman <- fvma/fvma[lg]                      # normalize
   sfun <- stepfun(ma1,c(0,fvman), f = 0)      # make it a step function
return(sfun)
}                                              # end step function
sfun <- stpfunc(db[[gene.sets[1]]])            # compute first step function
{plot(sfun,verticals = TRUE,do.points=FALSE,   # plot it
     xlim=c(0,1),ylim=c(0,1),col=colstep,
     xlab=xlab,ylab=ylab,
     main=title,cex.axis=cexa,cex.main=cexm)}
len <- length(gene.sets)                       # get number of plots
if (len>1){                                    # if more than one
 {lapply(gene.sets[2:len],                     # iterate
  function(x){lines(stpfunc(db[[x]]),xlim=c(0,1),
  verticals = TRUE,do.points=FALSE,col=4)})}
}
abline(0,1,lty=lty,lwd=lth,col=colline)        # add bisector
lg1 <- length(distr)
distrsh <- c(0,distr[1:(lg1-1)])               # F_{t_{i-1}}
diff <- distr-distrsh                          # differences F_{t_i}-F_{t_{i-1}}
```

```
intg <- cumsum(weights*diff)                  # calculate (approximate) the integral
                                              # (\int_0^t g(u)dF)
norm_fact <- intg[lg1]                        # normalization factor (\int_0^1 g(u)dF)
intgn <- intg/norm_fact                       # normalize it so that it sums up to 1
{lines(seq(1/lgw,1,1/lgw),intgn,
   lty=lty,lwd=lth,col=colline)}              # add the expected theoretical
}                                             # end function cumulated.weights


#-----------------------------------------------------------------
#          Distribution to be used for the DWKS test
#-----------------------------------------------------------------

#------------------ auxiliary functions --------------------------

group.frequencies <- function(db,nt){
#   computes a grouping of the gene frequencies in db
#   by clumping neighboring frequencies from empirical
#   distribution. The number of classes is nt.
#   Classes are clumped while the proportion is less
#   than hu (hurdle). The last class is clumped to the previous if
#   its proportion is less than hu/2. The hurdle is then adjusted
#   by dichotomy to get exacty nt classes.
#
Bflat <- db; names(Bflat)<-NULL     # remove names
Bflat <- unlist(Bflat)              # concatenate all pathways
tg <- table(Bflat)                  # different genes with frequencies
if (missing(nt)){                   # no type number specified
   nt <- nclass.Sturges(tg)         # use Sturges method
}
fg <- table(as.vector(tg))          # frequency distribution
pg <- as.integer(names(fg))         # values
ng <- as.integer(fg)                # numbers
to <- sum(ng)                       # total number
hu <- to/nt                         # hurdle
li <- length(pg)                    # last index
ci <- 0                             # current index
cn <- 0                             # current number
ct <- 0                             # current total
cc <- 0                             # current class
cp <- 0                             # current frequency
cL <- as.list(NULL)                 # initialize class list
cN <- NULL                          # initialize list of numbers
cP <- NULL                          # initialize vector of probas
nc <- 0                             # initialize count of classes
while (ct<to){                      # stop when all values are clumped
   while((cn<hu)&&(ci<li)){         # form a class larger than hurdle
      ci <- ci+1                    # increment current index
      cn <- cn+ng[ci]               # add new number
      cp <- cp+pg[ci]*ng[ci]        # cumulate frequencies
      cc <- c(cc,pg[ci])            # clump to current class
   }                                # end while
   nc <- nc+1                       # one more class
```

```
   cL[[nc]] <- cc                  # stack new class in class list
   ct <- ct+cn                     # increment number of clumped values
   cN <- c(cN,cn)                  # stack number
   cP <- c(cP,cp)                  # stack mean frequency
   cc <- NULL                      # empty current class
   cn <- 0                         # empty current number
   cp <- 0                         # empty current probability
}                                  # end while
if (cN[[nc]]<hu/2){                # last class weak
   ncm1 <- nc-1                    # next to last class
   cL[[ncm1]] <-{c(cL[[ncm1]],
                   cL[[nc]])}      # clump last two classes
   cN[ncm1] <-{cN[ncm1]+
                   cN[nc]}         # add last number to previous
   cP[ncm1] <-{cP[ncm1]+
                   cP[nc]}         # add last freq count to previous
   nc <- ncm1                      # new number of classes
   cL <- cL[1:nc]                  # remove last entry
   cN <- cN[1:nc]                  # remove last entry
   cP <- cP[1:nc]                  # remove last entry
}                                  # end if
cP <- (cP/cN)/length(Bflat)        # normalize probabilities
ty <- {lapply(1:nc,function(i)     # list of nc vectors
         {v <- {lapply(cL[[i]],    # for each stratum
                function(j)        # get gene names for each clumped class
                {names(which(tg==j))})}
          v<-unlist(v)
          return(v)})}
res <- {list(classes=cL,           # class definitions,
             counts=cN,            # class numbers
             probas=cP,            # inclusion probabilities
             strata=ty)}           # groups of genes of each stratum
return(res)
}                                  # end function group.frequencies


reduce.symbols <- function(pv,db){
#   Reduces database db to symbols present in vector pv.
#   Returns the reduced database.
#
rdb <- {lapply(db,function(v){          # apply to all gene sets
     return(find.matching(v,pv))})}     # reduce gene set to symbols in pv
l <- lapply(rdb,length)                 # new gene set lengths
rdb <- rdb[which(l>0)]                  # remove empty gene sets
return(rdb)
}                                       # end function reduce.symbols

#--------------------- main function -----------------------------

cumdistr <- function(w,db,nt,plotting=FALSE,title=""){
#   Takes a named vector w, a database db and an integer nt.
#   Groups the genes of db into (approximately) nt groups,
#   according to their frequency in the database. Then, the
```

```
#   distribution function F-to be used in the DWKS test-
#   is estimated based on this grouping
#   and the vector w.
#
weights <- sort(w,decreasing=TRUE) # sort weights in decreasing order
gt <- group.frequencies(db,nt=nt)  # group genes according to their
                                   # frequency in the database
ty <- gt$strata                    # list of genes by group
nty <- gt$counts                   # number of genes by group
prs <- gt$probas                   # inclusion probabilities
len <- length(nty)                 # actual number of groups
probas <- NULL
                                   # vector of genes, each with
for (j in 1:len){                  # its inclusion probability
pr <- rep(prs[j],nty[j])
names(pr) <- ty[[j]]
probas <- c(probas,pr)
}
df <- probas[names(weights)]       # density function
df[which(is.na(df))] <- 0          # assign inclusion probability 0 to
                                   # the genes present only in the vector
cdf <- cumsum(df)                  # cumulative distribution function
lg <- length(cdf)
cdf <- cdf/cdf[lg]                 # normalized
if (plotting){                     # plot the estimated distribution function
plot(seq(1/lg,1,1/lg),cdf,type="l",xlab="x",ylab="F(x)",main=title)
abline(0,1)
}
return(cdf)
}                                  # end function cumdistr
```

These are only scripts, and not R packages. The entries have not been protected and some functions may fail on extreme entries. We have tried to respect the spirit and scope of R but we have also focused on clarity and readability of the codes. There is certainly room for gain in precision and computing time. You are welcome to read and modify the code for your own usage, and get back to us for possible improvement.

Bibliography

[1] L. G. Acevedo, M. Bieda, R. Green, and P. J. Farnham. Analysis of the mechanisms mediating tumor-specific changes in gene expression in human liver tumors. *Cancer Res.*, 68(8):2641–2651, 2008.

[2] M. Ackermann and K. Strimmer. A general modular framework for gene set enrichment analysis. *BMC Bioinformatics*, 10:47, 2009.

[3] Affymetrix. *Statistical Algorithms Description Document*. Affymetrix, Santa Clara, CA, 2002.

[4] Affymetrix. *Expression Analysis Technical Manual*. Affymetrix, Santa Clara, CA, 2004.

[5] A. V. Antonov, T. Schmidt, Y. Wang, and H. W. Mewes. ProfCom: a web tool for profiling the complex functionality of gene groups identified from high-throughput data. *Nucleic Acids Res.*, 36(Web Server issue):W347–W351, 2008.

[6] S. M. Arfin, A. D. Long, E. T. Ito, L. Tolleri, et al. Global gene expression profiling in Escherichia coli K12. The effects of integration host factor. *J. Biol. Chem.*, 275(38):29672–29684, 2000.

[7] T. B. Arnold and J. W. Emerson. Nonparametric Goodness-of-Fit Tests for Discrete Null Distributions. *R Journal*, 3(2):34–39, 2011.

[8] P. Baldi and A. D. Long. A Bayesian framework for the analysis of microarray expression data: Regularized t-test and statistical inferences of gene changes. *Bioinformatics*, 17(6):509–519, 2001.

[9] D. A. Barbie, P. Tamayo, J. S. Boehm, S. Y. Kim, et al. Systematic RNA interference reveals that oncogenic KRAS-driven cancers require TBK1. *Nature*, 462(7269):108–112, 2009.

[10] J. Barretina, G. Caponigro, N. Stransky, K. Venkatesan, et al. The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, 483(7391):603–607, 2012.

[11] T. Beissbarth and T. P. Speed. GOstat: find statistically overrepresented Gene Ontologies within a group of genes. *Bioinformatics*, 20(9):1464–1465, 2004.

[12] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. R. Statist. Soc. B*, 57(1):289–300, 1995.

[13] Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Ann. Statist.*, 29(4):1165–1188, 2001.

[14] K. R. Boheler and M. D. Stern. The new role of SAGE in gene discovery. *Trends Biotechnol.*, 21(2):55–57, 2003.

[15] B. M. Bolstad, R. A. Irizarry, M. Åstrand, and T. P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–193, 2003.

[16] A. Boorsma, B. C. Foat, D. Vis, F. Klis, et al. T-profiler: scoring the activity of predefined groups of genes using gene expression data. *Nucleic Acids Res.*, 33(Web Server issue):W592–W595, 2005.

[17] B. Boukai. An explicit expression for the distribution of the supremum of Brownian motion with a change point. *Comm. Statist. Theory Methods*, 19:31–40, 1990.

[18] A. Brazma, H. Parkinson, U. Sarkans, M. Shojatalab, et al. ArrayExpress–a public repository for microarray gene expression data at the EBI. *Nucleic Acids Res.*, 31(1):68–71, 2003.

[19] R. Breitling, P. Armengaud, A. Amtmann, and P. Herzyk. Rank products: a simple, yet powerful, new method to detect differentially regulated genes in replicated microarray experiments. *FEBS Lett.*, 573(1-3):83–92, 2004.

[20] M. Carlson. *hgu133a.db: Affymetrix Human Genome U133 Set annotation data (chip hgu133a).* R package version 2.14.0.

[21] M. Carlson. *hgu133plus2.db: Affymetrix Human Genome U133 Plus 2.0 Array annotation data (chip hgu133plus2).* R package version 2.14.0.

[22] M. Carlson. *hgug4110b.db: Agilent Human 1A (V2) annotation data (chip hgug4110b).* R package version 2.14.0.

[23] M. Carlson. *org.Hs.eg.db: Genome wide annotation for Human.* R package version 2.14.0.

[24] P. Carmona-Saez, M. Chagoyen, F. Tirado, J. M. Carazo, et al. GENECODIS: a web-based tool for finding significant concurrent annotations in gene lists. *Genome Biol.*, 8(1):R3, 2007.

[25] U. Chandran. Introduction to microarray analysis. University Lecture, 2012.

[26] K. Charmpi and B. Ycart. Weighted Kolmogorov Smirnov testing: an alternative for Gene Set Enrichment Analysis. eprint arXiv:1410.1620, 2014.

[27] L. Chen, S. Wang, Y. Zhou, X. Wu, et al. Identification of early growth response protein 1 (EGR-1) as a novel target for JUN-induced apoptosis in multiple myeloma. *Blood*, 115(1):61–70, 2010.

[28] W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *JASA*, 74 (368):829–836, 1979.

[29] P. Clote and R. Backofen. *Computational Molecular Biology: An Introduction.* John Wiley & Sons, Ltd., Chichester, 2000.

[30] E. R. Cook and L. A. Kairiukstis. *Methods of Dendrochronology: Applications in the Environmental Sciences.* Kluwer Academic Publishers, Doredrecht, 1990.

[31] X. Cui and G. A. Churchill. Statistical tests for differential expression in cDNA microarray experiments. *Genome Biol.*, 4(4):210, 2003.

[32] S. Davis and P. Meltzer. GEOquery: a bridge between the Gene Expression Omnibus (GEO) and BioConductor. *Bioinformatics*, 23(14):1846–1847, 2007.

[33] M. A. Dawson and T. Kouzarides. Cancer epigenetics: from mechanism to therapy. *Cell*, 150(1):12–27, 2012.

[34] H. J. de Jonge, P. J. Valk, N. J. Veeger, A. ter Elst, et al. High VEGFC expression is associated with unique gene expression profiles and predicts adverse prognosis in pediatric and adult acute myeloid leukemia. *Blood*, 116(10):1747–1754, 2010.

[35] I. Dinu, Q. Liu, J. D. Potter, A. J. Adewale, et al. A biological evaluation of six gene set analysis methods for identification of differentially expressed pathways in microarray data. *Cancer Inform.*, 6:357–368, 2008.

[36] I. Dinu, J. D. Potter, T. Mueller, Q. Liu, et al. Improving gene set analysis of microarray data by SAM-GS. *BMC Bioinformatics*, 8:242, 2007.

[37] S. Drăghici, P. Khatri, R. P. Martins, G. C. Ostermeier, et al. Global functional profiling of gene expression. *Genomics*, 81(2):98–104, 2003.

[38] A. Drawid, R. Jansen, and M. Gerstein. Genome-wide analysis relating expression level with protein subcellular localization. *Trends Genet.*, 16(10):426–430, 2000.

[39] S. Dudoit, R. Gentleman, R. Irizarry, and Y. H. Yang. Pre-processing in cDNA microarray experiments. University Lecture, `http://homes.di.unimi.it/~valenti/BF1/SlideCorso/PreProc-cDNA.pdf`.

[40] S. Dudoit and M. van der Laan. *Multiple Testing Procedures with Applications to Genomics*. Springer, New York, 2007.

[41] M. Dunning, A. Lynch, and M. Eldridge. *illuminaHumanv3.db: Illumina HumanHT12v3 annotation data (chip illuminaHumanv3)*. R package version 1.22.1.

[42] M. Dunning, A. Lynch, and M. Eldridge. *illuminaHumanv4.db: Illumina HumanHT12v4 annotation data (chip illuminaHumanv4)*. R package version 1.22.1.

[43] R. Edgar, M. Domrachev, and A. E. Lash. Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Res.*, 30(1):207–210, 2002.

[44] S. Falcon and R. Gentleman. Using GOstats to test gene lists for GO term association. *Bioinformatics*, 23(2):257–258, 2007.

[45] K. Fendick. Gaussian fluid queue with autocorrelated input. eprint arXiv:1104.4741, 2011.

[46] A. Fog. Calculation Methods for Wallenius' Noncentral Hypergeometric Distribution. *Comm. Statist. Simulation Comput.*, 37(2):258–273, 2008.

[47] E. Frei, C. Visco, Z. Y. Xu-Monette, S. Dirnhofer, et al. Addition of rituximab to chemotherapy overcomes the negative prognostic impact of cyclin E expression in diffuse large B-cell lymphoma. *J. Clin. Pathol.*, 66(11):956–961, 2013.

[48] B. L. Fridley, G. D. Jenkins, and J. M. Biernacka. Self-contained gene-set analysis of expression data: An evaluation of existing and novel methods. *PLoS ONE*, 5(9):e12693, 2010.

[49] L. Gautier, L. Cope, B. M. Bolstad, and R. A. Irizarry. affy–analysis of Affymetrix GeneChip data at the probe level. *Bioinformatics*, 20(3):307–315, 2004.

[50] R. C. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, et al. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol.*, 5(10):R80, 2004.

[51] J. D. Gibbons and S. Chakraborti. *Nonparametric statistical inference*. Dekker, Basel, 2003.

[52] J. J. Goeman and P. Bühlmann. Analyzing gene expression data in terms of gene sets: methodological issues. *Bioinformatics*, 23(8):980–987, 2007.

[53] A. Heider and R. Alt. virtualArray: a R/bioconductor package to merge raw data from different microarray platforms. *BMC Bioinformatics*, 14:75, R package version 1.8.0, 2013.

[54] S. Héritier, E. Cantoni, S. Copt, and M. P. Victoria-Cantoni. *Robust methods in biostatistics*. Wiley, New York, 2009.

[55] D. G. Hernandez, M. A. Nalls, M. Moore, S. Chong, et al. Integration of GWAS SNPs and tissue specific expression profiling reveal discrete eQTLs for human traits in blood and brain. *Neurobiol. Dis.*, 47(1):20–28, 2012.

[56] J. I. Herschkowitz, K. Simin, V. J. Weigman, I. Mikaelian, et al. Identification of conserved gene expression features between murine mammary carcinoma models and human breast tumors. *Genome Biol.*, 8(5):R76, 2007.

[57] M. G. Hollingshead, L. H. Stockwin, S. Y. Alcoser, D. L. Newton, et al. Gene expression profiling of 49 human tumor xenografts from in vitro culture through multiple in vivo passages–strategies for data mining in support of therapeutic studies. *BMC Genomics*, 15(1):393, 2014.

[58] D. W. Huang, B. T. Sherman, and R. A. Lempicki. Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. *Nucleic Acids Res.*, 37(1):1–13, 2009.

[59] D. W. Huang, B. T. Sherman, and R. A. Lempicki. Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. *Nat. Protoc.*, 4(1):44–57, 2009.

[60] D. W. Huang, B. T. Sherman, Q. Tan, J. Kir, et al. DAVID bioinformatics resources: expanded annotation database and novel algorithms to better extract biology from large gene lists. *Nucleic Acids Res.*, 35(Web Server issue):W169–W175, 2007.

[61] J. H. Hung, T. H. Yang, Z. Hu, Z. Weng, et al. Gene set enrichment analysis: performance evaluation and usage guidelines. *Brief. Bioinform.*, 13(3):281–291, 2012.

[62] Illumina. *Array-Based Gene Expression Analysis*. Illumina, Inc., San Diego, CA, 2011.

[63] R. A. Irizarry, B. Hobbs, F. Collin, Y. D. Beazer-Barclay, et al. Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, 4(2):249–264, 2003.

[64] R. A. Irizarry, C. Wang, Y. Zhou, and T. P. Speed. Gene set enrichment analysis made simple. *Stat. Methods Med. Res.*, 18(6):565–575, 2009.

[65] R. A. Irizarry and Z. Wu with contributions from S. Cawley. *affycomp: Graphics Toolbox for Assessment of Affymetrix Expression Measures*. R package version 1.40.0.

[66] A. K. Järvinen, S. Hautaniemi, H. Edgren, P. Auvinen, et al. Are data from different gene expression microarray platforms comparable? *Genomics*, 83(6):1164–1168, 2004.

[67] M. A. Kayala and P. Baldi. Cyber-T web server: differential analysis of high-throughput data. *Nucleic Acids Res.*, 40(W1):W553–W559, 2012.

[68] P. Khatri and S. Drăghici. Ontological analysis of gene expression data: current tools, limitations, and open problems. *Bioinformatics*, 21(18):3587–3595, 2005.

[69] P. Khatri, S. Drăghici, G. C. Ostermeier, and S. A. Krawetz. Profiling gene expression using onto-express. *Genomics*, 79(2):266–270, 2002.

[70] S. Y. Kim and D. J. Volsky. PAGE: Parametric analysis of gene set enrichment. *BMC Bioinformatics*, 6:144, 2005.

[71] M. R. Kosorok. *Introduction to Empirical Processes and Semiparametric Inference*. Springer, New York, 2008.

[72] W. P. Kuo, T. K. Jenssen, A. J. Butte, L. Ohno-Machado, et al. Analysis of matched mRNA measurements from two different microarray technologies. *Bioinformatics*, 18(3):405–412, 2002.

[73] J. Lamb, E. D. Crawford, D. Peck, J. W. Modell, et al. The connectivity map: using gene-expression signatures to connect small molecules, genes, and disease. *Science*, 313(5795):1929–1935, 2006.

[74] C. Laurent, K. Charmpi, P. Gravelle, M. Tosolini, et al. Several immune escape patterns in non-Hodgkin's lymphomas. *to appear in OncoImmunology*, 2015.

[75] C. Li and W. H. Wong. Model-based analysis of oligonucleotide arrays: expression index computation and outlier detection. *PNAS*, 98(1):31–36, 2001.

[76] Y. Liao, G. K. Smyth, and W. Shi. The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. *Nucleic Acids Res.*, 41(10):e108, 2013.

[77] H. Y. Lim, I. Sohn, S. Deng, J. Lee, et al. Prediction of disease-free survival in hepatocellular carcinoma by gene expression profiling. *Ann. Surg. Oncol.*, 20(12):3747–3753, 2013.

[78] Q. Liu, I. Dinu, A. J. Adewale, J. D. Potter, et al. Comparative evaluation of gene-set analysis methods. *BMC Bioinformatics*, 8:431, 2007.

[79] D. J. Lockhart, H. Dong, M. C. Byrne, M. T. Follettie, et al. Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nat. Biotechnol.*, 14(13):1675–1680, 1996.

[80] I. Lönnstedt and T. Speed. Replicated microarray data. *Statist. Sinica*, 12:31–46, 2002.

[81] W. Luo, M. S. Friedman, K. Shedden, K. D. Hankenson, et al. GAGE: generally applicable gene set enrichment for pathway analysis. *BMC Bioinformatics*, 10:161, 2009.

[82] H. Maciejewski. Gene set analysis methods: statistical models and methodological differences. *Brief. Bioinform.*, 15(4):504–518, 2014.

[83] N. Mah, A. Thelin, T. Lu, S. Nikolaus, et al. A comparison of oligonucleotide and cDNA-based microarray systems. *Physiol. Genomics*, 16(3):361–370, 2004.

[84] M. Z. Man, X. Wang, and Y. Wang. POWER_SAGE: comparing statistical tests for SAGE experiments. *Bioinformatics*, 16(11):953–959, 2000.

[85] L. Marisa, A. de Reyniès, A. Duval, J. Selves, et al. Gene expression classification of colon cancer into molecular subtypes: characterization, validation, and prognostic value. *PLoS Med.*, 10(5):e1001453, 2013.

[86] J. Mayerle, C. M. den Hoed, C. Schurmann, L. Stolk, et al. Identification of genetic loci associated with Helicobacter pylori serologic status. *JAMA*, 309(18):1912–1920, 2013.

[87] A. M. Mikheev, T. Nabekura, A. Kaddoumi, T.K. Bammler, et al. Profiling gene expression in human placentae of different gestational ages: an OPRU Network and UW SCOR Study. *Reprod. Sci.*, 15(9):866–877, 2008.

[88] K. I. Mills, A. Kohlmann, P. M. Williams, L. Wieczorek, et al. Microarray-based classifiers and prognosis models identify subgroups with distinct clinical outcomes and high risk of AML transformation of myelodysplastic syndrome. *Blood*, 114(5):1063–1072, 2009.

[89] V. K. Mootha, C. M. Lindgren, K. F. Eriksson, A. Subramanian, et al. PGC-1alpha-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes. *Nat. Genet.*, 34:267–273, 2003.

[90] D. Nam and S. Y. Kim. Gene-set approach for expression pattern analysis. *Brief. Bioinform.*, 9(3):189–197, 2008.

[91] G. Obermoser, S. Presnell, K. Domico, H. Xu, et al. Systems scale interactive exploration reveals quantitative and qualitative differences in response to influenza and pneumococcal vaccines. *Immunity*, 38(4):831–844, 2013.

[92] F. Pont, M. Tosolini, B. Ycart, and J. J. Fournié. nwCompare and AutoCompare Softwares for Proteomics and Transcriptomics Data Mining - Application to the Exploration of Gene Expression Profiles of Aggressive Lymphomas. *Integrative Proteomics, H.C. Leung ed., InTech open access publisher ISBN 978-953-51-0070-6*, 2012.

[93] V. Popovici, W. Chen, B. G. Gallas, C. Hatzis, et al. Effect of training-sample size and classification difficulty on the accuracy of genomic predictors. *Breast Cancer Res.*, 12(1):R5, 2010.

[94] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. ISBN 3-900051-07-0.

[95] I. Rivals, L. Personnaz, L. Taing, and M. C. Potier. Enrichment or depletion of a GO category within a class of genes: which test? *Bioinformatics*, 23(4):401–407, 2007.

[96] R. Roth. Gene expression dataset for GSE7307. `http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE7307`, 2007.

[97] T. Sauer. Computational solution of stochastic differential equations. *WIREs Comput. Stat.*, 5(5):362–371, 2013.

[98] M. Schena, D. Shalon, R. W. Davis, and P. O. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270(5235):467–470, 1995.

[99] M. Schena, D. Shalon, R. Heller, A. Chai, et al. Parallel human genome analysis: microarray-based expression monitoring of 1000 genes. *PNAS*, 93(20):10614–10619, 1996.

[100] J. Seok, H. S. Warren, A. G. Cuenca, M. N. Mindrinos, et al. Genomic responses in mouse models poorly mimic human inflammatory diseases. *PNAS*, 110(9):3507–3512, 2013.

[101] I. Shmulevich and E. R. Dougherty. *Genomic Signal Processing*. Princeton University Press, 2007.

[102] G. R. Shorack and J. A. Wellner. *Empirical Processes with Applications to Statistics*. John Wiley & Sons Inc., New York, 1986.

[103] G. Smyth and T. Speed. Normalization of cDNA microarray data. *Methods*, 31(4):265–273, 2003.

[104] G. K. Smyth. Limma: linear models for microarray data. In R. Gentleman, V. Carey, S. Dudoit, R. Irizarry, and W. Huber, editors, *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, pages 397–420. Springer, New York, 2005.

[105] S. Song and M. A. Black. Microarray-based gene set analysis: a comparison of current methods. *BMC Bioinformatics*, 9:502, 2008.

[106] P. Stafford. *Methods in Microarray Normalization*. CRC Press, USA, 2008.

[107] A. Subramanian, H. Kuehn, J. Gould, P. Tamayo, et al. GSEA-P: a desktop application for Gene Set Enrichment Analysis. *Bioinformatics*, 23(23):3251–3253, 2007.

[108] A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, et al. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *PNAS*, 102(43):15545–15550, 2005.

[109] P. Tamayo and D. Eby. Ranknormalize documentation. `ftp://ftp.broadinstitute.org/pub/genepattern/modules_public_server_doc/RankNormalize.pdf`.

[110] J. Taminau. *inSilicoMerging: Collection of Merging Techniques for Gene Expression Data*, R package version 1.8.0, 2014.

[111] J. Taminau, D. Steenhoff, A. Coletta, S. Meganck, et al. inSilicoDb: an R/Bioconductor package for accessing human Affymetrix expert-curated datasets from GEO. *Bioinformatics*, 27(22):3204–3205, 2011.

[112] T. S. Tanaka, S. A. Jaradat, M. K. Lim, G. J. Kargul, et al. Genome-wide expression profiling of mid-gestation placenta and embryo using a 15,000 mouse developmental cDNA microarray. *PNAS*, 97(16):9127–9132, 2000.

[113] A. L. Tarca, G. Bhatti, and R. Romero. A comparison of gene set analysis methods in terms of sensitivity, prioritization and specificity. *PLoS ONE*, 8(11):e79217, 2013.

[114] L. Tian, S. A. Greenberg, S. W. Kong, J. Altschuler, et al. Discovering statistically significant pathways in expression profiling studies. *PNAS*, 102(38):13544–13549, 2005.

[115] R. Tibshirani, G. Chu, B. Narasimhan, and J. Li. *samr: SAM: Significance Analysis of Microarrays*, R package version 2.0, 2011.

[116] A. Tsodikov, A. Szabo, and D. Jones. Adjustments and measures of differential expression for microarray data. *Bioinformatics*, 18(2):251–260, 2002.

[117] V. G. Tusher, R. Tibshirani, and G. Chu. Significance analysis of microarrays applied to the ionizing radiation response. *PNAS*, 98(9):5116–5121, 2001.

[118] P. Warnat, R. Eils, and B. Brors. Cross-platform analysis of cancer microarray data improves gene expression based classification of phenotypes. *BMC Bioinformatics*, 6:265, 2005.

[119] H. J. Westra, M. J. Peters, T. Esko, H. Yaghootkar, et al. Systematic identification of trans-eQTLs as putative drivers of known disease associations. *Nat. Genet.*, 45(10):1238–1243, 2013.

[120] A. R. Wood, D. G. Hernandez, M. A. Nalls, H. Yaghootkar, et al. Allelic heterogeneity and more detailed analyses of known loci explain additional phenotypic variation and reveal complex patterns of association. *Hum. Mol. Genet.*, 20(20):4082–4092, 2011.

[121] D. Wu and G. K. Smyth. Camera: a competitive gene set test accounting for inter-gene correlation. *Nucleic Acids Res.*, 40(17):e133, 2012.

[122] M. C. Wu and X. Lin. Prior biological knowledge-based approaches for the analysis of genome-wide expression profiles using gene sets and pathways. *Stat. Methods Med. Res.*, 18(6):577–593, 2009.

[123] Z. Wu, R. A. Irizarry, R. Gentleman, F. Martinez-Murillo, et al. A model-based background adjustment for oligonucleotide expression arrays. *JASA*, 99(468):909–917, 2004.

[124] W. Xiao, M. N. Mindrinos, J. Seok, J. Cuschieri, et al. A genomic storm in critically injured humans. *J. Exp. Med.*, 208(13):2581–2590, 2011.

[125] B. Ycart, K. Charmpi, S. Rousseaux, and J. J. Fournié. Large scale statistical analysis of GEO datasets. *Gene Technology*, 3(1):1–9, 2014.

[126] B. Ycart, F. Pont, and J. J. Fournié. Maximum entropy testing in Bioinformatics databases. Research report, 2012.

[127] B. Ycart, F. Pont, and J. J. Fournié. Checking false discovery rates on pvplots. *InterStat*, July #005, 2013.

[128] B. Ycart, F. Pont, and J. J. Fournié. Curbing false discovery rates in interpretation of genome-wide expression profiles. *J. Biomed. Inform.*, 47:58–61, 2014.

[129] J. Zhang, R. Chiodini, A. Badr, and G. Zhang. The impact of next-generation sequencing on genomics. *J. Genet. Genomics*, 38(3):95–109, 2011.

Méthodes statistiques pour la fouille de données dans les bases de données de génomique

Résumé:

Cette thèse est consacrée aux tests statistiques, visant à comparer un vecteur de données numériques, indicées par l'ensemble des gènes du génome humain, à un certain ensemble de gènes, connus pour être associés par exemple à un type donné de cancer. Parmi les méthodes existantes, le test Gene Set Enrichment Analysis est le plus utilisé. Néanmoins, il a deux inconvénients. D'une part, le calcul des p-valeurs est coûteux et peu précis. D'autre part, il déclare de nombreux résultats significatifs, dont une majorité n'ont pas de sens biologique. Ces deux problèmes sont traités, par l'introduction de deux procédures statistiques nouvelles, les tests de Kolmogorov-Smirnov pondéré et doublement pondéré. Ces deux tests ont été appliqués à des données simulées et réelles, et leurs résultats comparés aux procédures existantes. Notre conclusion est que, au-delà leurs avantages mathématiques et algorithmiques, les tests proposés pourraient se révéler, dans de nombreux cas, plus informatifs que le test GSEA classique, et traiter efficacement les deux problèmes qui ont motivé leur construction.

Statistical methods for data mining in genomics databases (Gene Set Enrichment Analysis)

Abstract:

Our focus is on statistical testing methods, that compare a given vector of numeric values, indexed by all genes in the human genome, to a given set of genes, known to be associated to a particular type of cancer for instance. Among existing methods, Gene Set Enrichment Analysis is the most widely used. However it has several drawbacks. Firstly, the calculation of p-values is very much time consuming, and insufficiently precise. Secondly, like most other methods, it outputs a large number of significant results, the majority of which are not biologically meaningful. The two issues are addressed here, by two new statistical procedures, the Weighted and Doubly Weighted Kolmogorov-Smirnov tests. The two tests have been applied both to simulated and real data, and compared with other existing procedures. Our conclusion is that, beyond their mathematical and algorithmic advantages, the WKS and DWKS tests could be more informative in many cases, than the classical GSEA test and efficiently address the issues that have led to their construction.