FREE UNIVERSITY OF BRUSSELS

Faculty Of Sciences

Master In Bioinformatics

Breast Cancer Diagnosis Using Microarray

Thesis presented for the
Master in Bioinformatics
by Benjamin Haibe-Kains

Training Master :
M. Christos Sotiriou

Training Supervisor :
M. Gianluca Bontempi

Academic Year 2003 - 2004

Contents

Acknowledgments

I wish to thank:

G. Bontempi who followed me since my licence in computer science and initiated the collaboration with Christos Sotiriou. His help was invaluable for my research and the redaction of this thesis.

Christos Sotiriou who allows me to do the training in his laboratory, his great help on numerous research fields and much more.

All my colleagues in the Microarray Unity at the IJB for their enthusiasm and their efficiency.

Jean-Francois Laes who allows me to follow a very good training at Heidelberg and his help in biology.

Philppe Hennebert for his crazy behavior during intensive work and his original ideas.

Chrisitine Desmedt, Francoise Lallemand and Virginie Durbecq for their great discussions and their motivation.

Valerie Huyge who allows me to clarify my knowledge in breast cancer diagnosis.

All the TransBIG team for the 20 km of Brussels (have you seen the "Think BIG !" t-shirts ? Too fast probably ...).

Sylvain Brohee and the others students who follows the master in Bioinformatics.

Finally all my teachers who have opened my mind to such interesting research fields.

I hope that my friends and my family have not really suffered from my bad mood during intensive work. Their support was not only valuable but necessary.

Chapter 1

Introduction

Thanks to the generalization of the screening mammogram, more patients are diagnosed with early breast cancer (small tumors and absence of lymph node invasion). Even if these patients achieve a long term survival, 20 to 30 percent of them will relapse and will die from their disease. The majority of these deaths is due to distant metastases. Loco-regional treatment (surgery and radiotherapy) is always carried out and a systemic adjuvant treatment (e.g. chemotherapy) is proposed to all high risk patients to prevent recurrence.

This risk is defined from several histological criteria (established during consensus conferences in Europe and USA [2, 16, 39, 1]) which constitutes the cancer "personality" [10]:

- Invasive/non-invasive breast cancer:
  - Non-invasive (or "in situ") cancers confine themselves to the ducts or lobules and do not spread to the surrounding tissues in the breast or other parts of the body. They can, however, develop into or increase your risk for invasive cancer.
  - Invasive (or infiltrating) cancers have started to break through normal breast tissue barriers and invade surrounding areas. Much more serious than non-invasive cancers, invasive cancers can spread to other parts of the body through the bloodstream and lymphatic system.
- Number of involved lymph nodes: some breast cancers spread to the lymph nodes under the arm. When the lymph nodes are involved in the cancer, they are called "positive". When lymph nodes are free of cancer, they are called "negative". In large medical studies, there appears to be a correlation between the number of involved lymph nodes and how aggressive a cancer's personality will be. Knowing how many lymph nodes are affected by cancer will help to select the appropriate treatment to fight the cancer.
- Tumor size.
- Tumor rate/grade:
  - Rate of cancer cell growth: the proportion of cancer cells growing and making new cells varies from tumor to tumor and may be helpful in predicting how aggressive a cancer is. If more than 6-10% of the cells are making new cells, the rate of growth is considered unfavorably high.
  - Grade of cancer cell growth: patterns of cell growth are rated on a scale from 1 to 3 (also referred to as low, medium, and high instead of 1, 2 or 3). Calm, well-organized growth with few cells reproducing is considered grade 1. Disorganized, irregular growth patterns in which many cells are in the process of making new cells is called grade 3. The lower the grade, the more favorable the expected outcome. At the same time, the higher the grade, the more vulnerable the cancer is to treatments such as chemotherapy and radiation.

  - Dead cells within the tumor: it is tempting to think that the only good cancer cell is a dead cancer cell. However, necrosis (or dead tumor cells) is one of several signs of excessive tumor growth.

- Hormone receptor status: estrogen and progesterone stimulate the growth of normal breast cells as well as some breast cancer cells. If a tumor is estrogen-receptor positive (ER-positive), it is more likely to grow in a high-estrogen environment. ER-negative tumors are usually not affected by the levels of estrogen and progesterone in your body. ER-positive cancers are more likely to respond to anti-estrogen therapies (tamoxifen, a drug that works by blocking the estrogen receptors on the breast tissue cells and slowing their estrogen-fueled growth).

- Oncogene over-expression: oncogene over-expression happens when an oncogene (such as HER2/neu, EGFR, and p53) over-expresses itself by making excess normal or abnormal proteins and receptors. Cancers that result from over-expressed oncogenes tend to be more nasty or belligerent and are more likely to recur than other cancers. They also may respond to different types of treatment than other breast cancers.

- Margins of resection: the term "margins" or "margins of resection" is used to refer to the distance between the tumor and the edge of the tissue taken by surgery. The margins are measured on all six sides: front and back, top and bottom, left and right.

According to these histological criteria, approximatively 80 percent of young patients without lymph node invasion are candidates for adjuvant treatment. It is obvious that these patients are over-treated because 70 to 80 percent of them will not develop distant metastases without the adjuvant treatment [17]. These results highlight the necessity to improve the risk evaluation based on traditional factors.

During last ten years, several prognosis factors (e.g. HER2 and p53 mutations) have been assessed and have been correlated to the prognosis but these genes, taken individually, have only a limited predictive power. This is probably due to the molecular complexity and heterogeneity of the tumors. The tumor phenotypes are not resumed by an oncogene up-regulation or an anti-oncogene mutation but can be the result of a series of genetic events. The tumor phenotype is not determined by isolated aberrations but by a combination of anomalies in a genetic context.

Currently, thanks to technology advances in genome sequencing, new tools are available to analyze biological experiments at the molecular level. The microarray technology allows to analyze the genetic identity of a specific tissue for the whole genome. In one microarray experiment, several thousands of gene expressions can be measured from a tumor tissue. This technology can be used to study the genetic context of the breast tumor to improve the risk evaluation and our understanding of this biological phenomenon.

1.1 State of the Art

According to a common view, progression from a primary to a metastatic tumor is accompanied by the sequential acquisition of phenotype changes, thus allowing breast cancer cells (BCC) to invade, disseminate, and colonize distant sites. Nevertheless, most investigations have revealed that progression is not accompanied by major changes in marker expression or grade [29].

These observations suggest that the metastatic signature might already be present in the primary breast tumor, challenging the traditional model of metastasis, which specifies that most primary tumor cells have low metastatic potential, but rare cells within large primary tumors acquire metastatic capacity through somatic mutations.

From that perspective, Van't Veer *et al.*, applying a supervised learning method [15], sought to identify whether there exists a gene expression signature strongly predictive of a short interval to distant metastases in primary breast cancer tumors [26]. For that purpose the authors investigated a narrow subset of breast cancer patients: node negative breast cancer patients, all under 55 years of age treated only with local regional therapies. They found 231 genes significantly associated

with disease outcome as defined by the presence of distant metastasis at the 5-year mark. They could then subsequently collapse this list into a core set of 70 prognostic markers. Interestingly, the investigators tested the ability of this array-derived prognostic "expression profile" to correctly identify patients who would need adjuvant chemotherapy and compared it to accepted guidelines for treatment of node negative breast cancer (NIH [16] and St. Gallen [2] consensus guidelines). They found that although the expression profile could correctly identify patients who would need adjuvant chemotherapy, it could effectively reduce the fraction of women not needing adjuvant chemotherapy by about 30%. The same Dutch group applied this signature to a larger test set of node negative and node positive breast cancer patients (295) from the same institution, who had been followed for 7 years. This study confirmed that the 70-gene prognosis signature could clearly distinguish patients with excellent 10 year survival from those with a high mortality rate [36].

These results suggest that molecular profiling might be able to substantially refine cancer prognosis, perhaps well beyond what is possible with other clinical indicators. All these molecular signatures might be generalizable to populations other than those in which they were initially developed, and probably across multiple microarray platforms and technologies. These hypotheses, however, need to be validated prior to implementation of these molecular signatures in the clinic. Finally, the molecular signatures of breast cancer cells captured by microarray technology suggest that the potential of breast cancer cells to metastasize may be genetically already determined earlier in the disease and be tissue-specific.

Several reports described the use of microarrays to assess a molecular classification of human breast cancers and investigated the possibility to correlate gene expression profiles with clinical outcome (see table 1.1). A brief introduction to each study is given in [11].

The supervised learning method used in microarray classification is illustrated in the figure 1.1: the learning method constructs a classifier in the basis of the microarray data (gene expressions) and the histological criterion (e.g. binary class representing the appearance of distant metastases in the first 5 years of follow-up). This classifier can be used to predict the class of new data (e.g. a tumor tissue from a new patient).

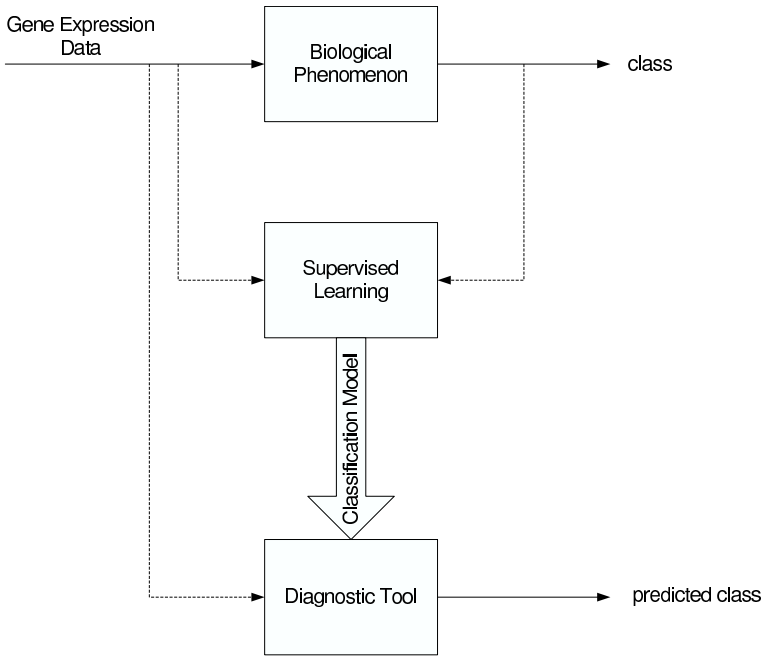


Figure 1.1: Supervised learning method used in microarray classification.

Study	**Platform**	**Patient cohort**	**Type of analysis**	**Informative genes**	**Findings**
Perou *et al.*, 2000 [42]	cDNA 8102 features	42 patients with normal or malignant breast tissues: 36 infiltrating ductal carcinomas, 2 lobular, 1 DCIS, 1 fibroadenoma and 3 normal breasts. 20 pairs of tumors were sampled before and after 16 weeks of doxorubicin chemotherapy.	Unsupervised	"intrinsic" gene subset 496 genes	Identification of 4 subgroups: -ER+/luminal-like -Basal-like -erbB2+ -Normal breast-like
Sorlie *et al.*, 2001 [52]	cDNA 8102 features	7 non-malignant breast samples and 78 carcinomas (71 ductal, 5 lobular, 2 DCIS- including 40 tumors from [42]).	Unsupervised	"intrinsic" gene subset [42]	(1) Identification of novel luminal-type sub-classes: A, B, C (2) Luminal A had the best clinical outcome
Sorlie *et al.*, 2003 [51]	cDNA 8102 features	Population [52] + 38 additional carcinomas.	Unsupervised	"intrinsic" gene subset [42]	Validation of [52]
Gruvberger *et al.*, 2001 [18]	cDNA 6728 features	58 node-negative breast tumors: 28 ER+ and 30 ER-	Supervised	Top 100 ER-discriminating genes	Discrimination between ER+ and ER- tumors
West *et al.*, 2001 [32]	Oligonucleotides 5600 genes	49 primary breast tumors: 13 ER+/LN+, 12 ER-/LN+, 12 ER+/LN-, 12 ER-/LN-	Supervised	(1) Top 100 ER-discriminating genes (2) Top 100 LN-discriminating genes	(1) Discrimination between ER+ and ER- tumors (2) Discrimination between LN+ and LN- tumors
Sotiriou *et al.*, 2003 [53]	cDNA 7650 features	99 invasive ductal carcinomas treated with adjuvant treatment: 53 LN+ and 46 LN-.	Unsupervised	706 genes	(1) Identification of 6 subgroups: -Luminal 1,2 and 3 -Basal-like 1 and 2 -erbB2+ (2) Luminal 1 had the best clinical outcome
Van't Veer *et al.*, 2002 [26]	Oligonucleotides 24479 genes	98 tumors: 78 sporadic tumors LN- from untreated patients (34 metastasis in 5 years, 44 disease free), 18 BRCA1 and 2 mutations carriers	Supervised	"intrinsic" gene subset	Prediction of clinical outcome, defined as the presence of metastases at the 5-year mark.
Van de Vijver *et al.*, 2002 [36]	Oligonucleotides 24479 genes	295 stage I or II breast cancer patients, both treated and untreated: 151 LN- and 144 LN+	Supervised	"prognosis" signature [26]	Validation of [26]

Table 1.1: Microarray studies in breast cancer [11].

1.2 TransBIG Project

The validation of all these promising results in larger, independent and, if possible, prospective series is urgently needed. With respect to the potential prognostic value of a breast cancer gene signature, the EORTC-BCG group is working to launch such a prospective validation trial with the support of the Breast International Group (BIG) and a grant from the European Union. This study, called TransBIG project, is designed to compare (in 5000 women with node negative breast cancer) treatment selection on the basis of classical prognostic/predictive factors (St-Gallen 2003 Guidelines, 2500 patients) or by the expression signature of the 70 genes identified by the Amsterdam group (2500 patients). While it is anticipated that the clinical outcome will be similar for the two groups, it is thought that the need for adjuvant chemotherapy will be reduced by 10 to 20 % in the group managed according to their tumor gene expression profile. Since ongoing research might show that gene expression arrays are of value in selecting optimal treatment at the time of relapse, our plan is to try to offer this technology to all women participating in the trial, but with a delayed potential use in the "control" group.

Another aim of this study is the development of a new signature based on the Affymetrix microarray platform. This part of the TransBIG project is currently supervised by Christos Sotiriou's laboratory at Jules Bordet Institute in collaboration with the SIB for the bioinformatics analysis [28]. My training concerns the development of this new signature, involving the quality assessment and supervised classification of breast tumors.

1.3 Contributions

Computer resources Application server installation to carry out large bioinformatics analyzes.

Quality assessment Implementation of R functions to assess quality of Affymetrix chips from *CEL* files following Affymetrix [5] and Bioconductor [14] guidelines.

Classification Implementation of R functions to perform a complete classification procedure (structural identification, feature selection, classification, validation).

Affymetrix data management Implementation of R functions to facilitate the management and the analysis of Affymetrix data (read data, preprocessing data)

Gene Ontology Use of GO tools (e.g. Onto-Express [47] or gominer [60]) to interpret biological results. Collaborations in [31, 22].

1.4 Glossary

Expressed Sequence Tag A short strand of DNA that is a part of a cDNA molecule and can act as identifier of a gene.

Gene Expression Transcription of the information contained within the DNA into messenger RNA (mRNA) molecules that are then translated into proteins.

Oligonucleotide Short fragment of a single-stranded DNA.

Polymerase Chain Reaction (PCR) Exponential amplification of almost any region of a selected DNA molecule.

Probe Easily detectable molecule which has the property to be located specifically either on another molecule, or in a given cellular compartment. Various molecules can be used as probe with condition that a marker (enzyme, compound radioactive or fluorescent) can be associated with the probe which allows its detection. Generally the probe is a nucleic acid fragment (ARN or ADN).

Meta-analysis Analysis involving several sources of microarray data (e.g. Affymetrix and Agilent data).

1.5 Abbreviations and Acronyms

BCC Breast Cancer Cell.

BIG Breast International Group.

CEL CELl intensities.

DCIS Ductal Carcinoma In Situ.

EORTC-BCG Breast Cancer Group of the European Organization for Research and Treatment in Cancer.

ER Estrogen Receptor.

EST Expressed Sequence Tag.

FN False Negatives.

FP False Positives.

GO Gene Ontology.

IGR Institute Gustave Roussy.

IJB Institute Jules Bordet.

JRH John Radcliffe Hospital.

Karolinska19 19 biological samples hybridized at the IJB and coming from Karolinska.

Karolinska68 68 *CEL* files coming from Karolinska and hybridized at the Karolinska Institute.

KNN K-Nearest Neighbours.

LN Lymph Node.

L-O-O Leave-One-Out.

MAS Microarray Affymetrix Suite.

MGED Microarray Gene Expression Data Society.

MIAME Minimum Information About a Microarray Experiment.

MM MisMatch.

PM Perfect Match.

RMA Robust Multi-array Average expression measure.

RT-PCR Reverse Transcriptase Polymerase Chain Reaction.

SIB Swiss Institute of Bioinformatics.

SVM Support Vector Machines.

TN True Negatives.

TP True Positives.

Chapter 2

Materials and Methods

2.1 Populations

Three different populations of patients are involved in the TransBig project:

- John Radcliffe Hospital (JRH), Oxford, UK (Dr Adrian Harris)
- Gustave Roussy Institute, Villejuif, France (Dr Suzette Delaloge)
- Karolinska Institute and Hospital (Karolinska), Stockholm, Sweden (Dr Jonas Bergh)

All the patients had tumors without lymph nodes invasion, were under 60 years old and have been not treated by adjuvant treatment (e.g. chemotherapy). The patients have to be followed up during five years (relapse during the first five years or relapse free during at least five years). The prognosis concerns the distant metastases. Some patients have had to be discarded because of the type of the relapse (loco-regional relapse instead of distant relapse), the insufficient follow-up and the lack of RNA amount.

The Microarray Unity of Institute Jules Bordet (IJB) have hybridized all the Affymetrix chips for the JRH and the IGR populations. At the beginning of 2004, BioVallee [58] have hybridized the Affymetrix chips for the Karolinska population with the same Affymetrix devices. For all the RNA experiments, the chips hgu133a and hgu133b (see section 2.2.1) have been hybridized.

2.2 Microarray Platform

Microarray technology is a powerful tool for genetic research that uses nucleic acid hybridization techniques to evaluate the mRNA expression profile of thousands of genes within a single experiment. The Microarray Unity of the IJB uses the Affymetrix platform [4] which is a short oligonucleotide platform (see appendix A for an overview of different microarray platforms). A part of Affymetrix devices can be seen in the figure 2.1.

2.2.1 Affymetrix Technology

Affymetrix chips are short oligonucleotide (25 mers) arrays fabricated by direct synthesis of oligonucleotides on the glass surface. Each chip contains up to 400,000 different oligos (called probes, see figure 2.2). Since oligonucleotide probes are synthesized in known locations on the chip, the hybridization pattern and signal intensities can be interpreted in terms of gene identity and relative expression levels by a specific software[1]. Each gene is represented on the chip by a series of different oligonucleotide probes (see figure 2.3). Each probe pair consists of a perfect

[1] We can mention the official Affymetrix Microarray Suite Software [4] or the Bioconductor [14] packages for R [43].

Figure 2.1: Affymetrix fluidic station and scanner [24].

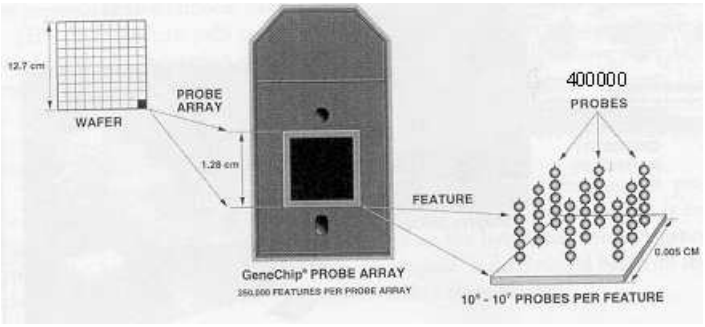


Figure 2.2: Manufacturing GeneChip probe array [24].

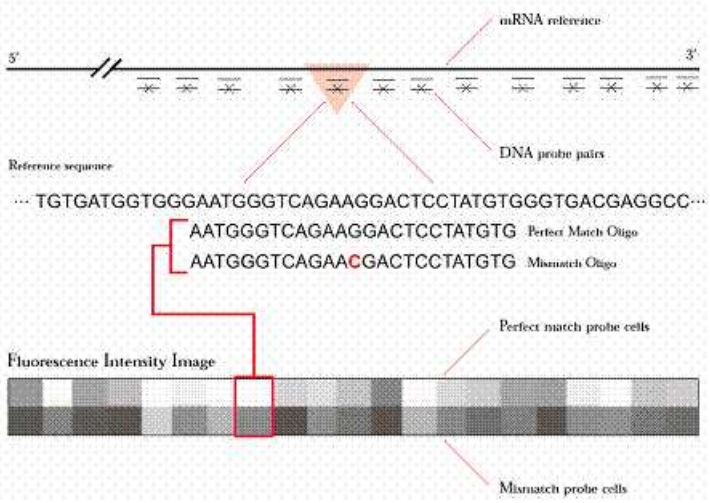


Figure 2.3: Oligonucleotide probe pair [24].

match (called PM) and a mismatch (called MM) oligonucleotide. The perfect match has a sequence exactly complementary to the particular region of gene and thus the probe set measures the expression of the gene. The mismatch probe differs from the perfect match probe by a single base substitution at the center base position, disturbing the bonding of the target gene transcript. This helps to determine the background and nonspecific hybridization that contributes to the signal measured for the perfect match oligo. Probes are chosen based on current information from GenBank and other nucleotide repositories. The sequences are believed to recognize unique regions of the 3' end of the gene.

Affymetrix Chips It exists several kind of Affymetrix chips for human: hgu95a, hgu95b, hgu133a, hgu133b, hgu133+, etc. For each patient the chips hgu133a (22283 affy ids[2]) and hgu133b (22645 affy ids) are used. The majority of known genes are on the chip hgu133a but the chip hgu133b is also used for the completeness (we hope to cover the whole human genome)

2.3 Quality Assessment

Quality assessment of the hybridized chips is an important step, often skipped in the analysis design.

The quality assessment can be divided in two steps:

1. Before the hybridization: some tests are carried out in the laboratory concerning e.g. mRNA quality and tissue purity.
2. After the hybridization: the quality control is based on information contained in the CEL files to assess the chip quality. There is no quality standard for the Affymetrix chip quality although some information can be found in the Affymetrix technical manual [5] and in studies from other laboratories [38].

To assess chip quality, several quality controls have been mixed, following the guidelines proposed by Affymetrix and Bioconductor [14].

Probe Array Image The chips are displayed in gray scale images. The gray intensities are computed on the basis of the CEL files. By inspecting these images, some artifacts can be discovered (e.g. broken area and bad chip washing).

Average Background Depending to the method, a background information can be computed. The Affymetrix guidelines suggest to compare the average background intensities (computed by the MAS 5.0 method, see appendix B.2.1) along all the chips (with a maximum value of 100).

Spike Controls and RNA Degradation Because RNA degradation typically starts from the 5' end of the molecule, we would expect probe intensities to be systematically lowered at that end of the probe set when compared to the 3' end. Affymetrix chips use probes at the 3' and 5' ends of the GAPDH and beta actin [27] genes to measure the RNA quality. A high 3' to M or 3' to 5' ratio, i.e. more than 3, may indicate degraded RNA or inefficient transcription of cDNA or biotinylated cRNA. Moreover additional probes are spiked in during the latter stages of the sample preparation process and are used to assess hybridization efficiency [35, 23]. These probes (BioB, BioC, BioD and CreX) should be detected as present (see detection calls).

[2]The majority of affy ids represent human genes but several are used for control or represent large region of transcribed DNA (EST).

Detection Calls A detection algorithm, developed by Affymetrix [5], uses probe pair intensities to generate detection p-value and assign a *Present*, *Marginal*, or *Absent* call. Each probe pair in a probe set is considered as having a potential vote in determining whether the measured transcript is detected (Present) or not detected (Absent). The vote is described by a value called the *discrimination score* (R). The score is calculated for each probe pair and is compared to a predefined threshold *Tau*. Probe pairs with scores higher than *Tau* vote for the presence of the transcript and inversely. The voting result is summarized as a *p-value* associated with the test of the difference between score and *Tau*.

The *discrimination score* is a basic property of a probe pair that describes its ability to detect its attended target (see figure 2.4):

$$R = (pm - mm)/(pm + mm)$$

where *pm* and *mm* are the values of the PM and the MM intensity of a probe pair.

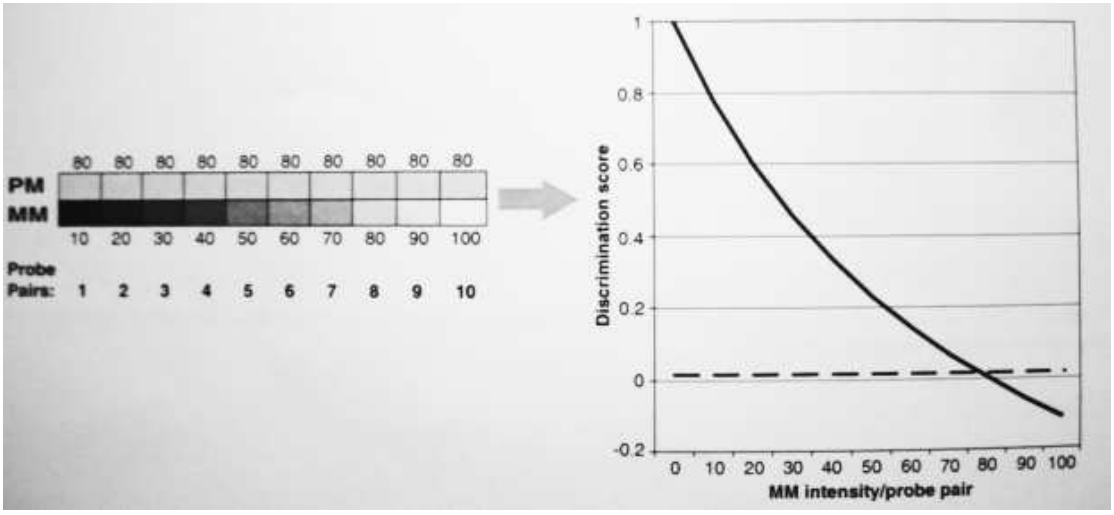


Figure 2.4: Discrimination factor [5]. The PM intensity is fixed to 80 and the MM intensity varies from 10 to 100. The y-axis represents the *discriminant score* and the x-axis represents the MM intensity.

Each *discrimination score* is compared to the threshold *Tau*. *Tau* is a small positive number[3] that can be adjusted to increase or decrease sensitivity and/or specificity of the analysis. Detection p-value is calculated by the One-Sided Wilcoxon's Signed Rank test. Finally, a detection call (*Present/Marginal/Absent*) is assigned for each probe set according to its detection p-value (see figure 2.5).

Extremely low values of *Present* calls percentage in the whole chips are a possible indication of poor sample quality.

Scaling Factor Since the majority of transcripts are not changing[4] among experiments, the overall intensities of the chips should be similar. Differences in overall intensity are most likely due to assay variables including pipetting error, hybridization, washing, and staining efficiencies, which are all independent of relative transcript concentration.

The scaling factor is used to assess the differences in overall intensity among experiments: it is the factor allowing to align the mean intensity of one chip on a target value. If the same

[3]Default value equals to 0.015.

[4]It is a strong hypothesis supported by the fact that each chip contains about 22,000 transcripts. The new chip hgu133+, which contains the whole human genome, is less sensible to this hypothesis.

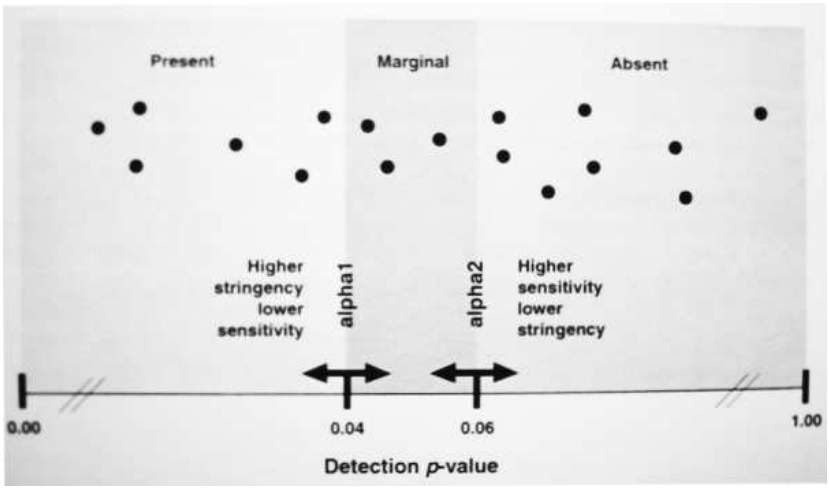


Figure 2.5: Detection p-value [5].

target value is used among all the experiments, the different factors should be similar: large discrepancies among scaling factors (e.g. three-fold or greater) may indicate significant assay variability or experiment degradation leading to noisier data.

Box Plots for PM Intensities Box plot of PM intensities can be useful to detect outliers, experiments for which the median and the interquartile are significantly different. After normalization, the boxes are typically well aligned. Rigorous statistical test could be implemented to test the difference between medians and interquartile intervals along all the experiments.

All these quality criteria are resumed in the table 2.1.

Quality criteria	**Guidelines**
Probe array image	Visual inspection of the array image. Research of artifacts.
Average background	Values should be similar between experiments and should not exceed 100.
Spike controls and RNA degradation	(1) RNA degradation: Values should be similar between experiments and should not exceed the interval [0.33, 3] (i.e. a ratio of 3). (2) Spike controls: 100% of the spike controls should be detected as present.
Detection calls	Values should be similar between experiments, should be close to 50% for the Present/Absent calls and should not exceed 10% for Marginal call.
Scaling factor	Values should be similar between experiments and should not exceed the interval [0.33, 3] (i.e. a ratio of 3).
box plots for the PM intensities	Detection of outlier and observation of the effect of normalization.

Table 2.1: Quality criteria and their acceptable values. This criteria are inspired by the Affymetrix and Bioconductor guidelines.

2.4 Supervised Classification

The studied diagnosis concerns the appearance of distant metastases during the first five years of follow-up. If such metastases exist, the patient belongs to the class "relapse", otherwise to the class "non-relapse". The problem consists in a binary supervised classification. Unfortunately, the common predictors, based on histological criteria presented previously, fail to classify accurately the breast tumors. However the treatments (e.g. chemotherapy and hormonal therapy) need to be specialized according to the patient tumor type. It is the reason for which the scientific community tries to develop new predictors based on gene expression profile to select patients who would benefit from adjuvant therapy.

The goal is twice:

- Reduce significantly the number of patients who receive unnecessary adjuvant therapy and reduce the adverse side effects for the treated patient. Moreover such therapies are very expensive.
- Isolate involved genes in breast cancer to improve our understanding of biological phenomena.

2.4.1 Analysis Design

The "traditional" design of the supervised classification in microarray is composed by several steps[5]:

1. Read data: the raw data from scanner have to be read in an useful format (e.g. a matrix or a data frame).
2. Get gene expression measures: the raw data have to be transformed to get the expression measure for each probe set in the chip. In the case of Affymetrix data, this step can be divided into three parts:
   (a) Background correction: the signal intensity have to be corrected because the measured intensity is noised. Actually, even an empty position (without oligo) returns a low level intensity.
   (b) Normalization: the background corrected data have to be normalized to allow comparison between the experiments.
   (c) Probe specific correction: the information given by the mismatch intensity can be used to correct the perfect match intensity.
   (d) Summarization: a gene expression measure is obtained by computing an expression from all the previously corrected probe set values.
3. Prefiltering: the normalized data can be prefiltered by using arbitrary criteria independently on the outcome (e.g. minimum variance or fold-change).
4. Filtering: the normalized data can be filtered by using the outcome of each experiment (e.g. *wilcoxon test* or *student t-test*)
5. Classification: the normalized data are classified by using a specific technique (e.g. *linear discriminant analysis* or *support vector machine*).

It is necessary that such a design considers the validation of the classification (see figure 2.6). In this case, the input is the microarray data and the histological information relative to each experiments. The output is a set of genes (called marker genes) which are used by the classifier. This set is called a marker gene signature.

[5]The details about all the analysis functions are given in appendix B.

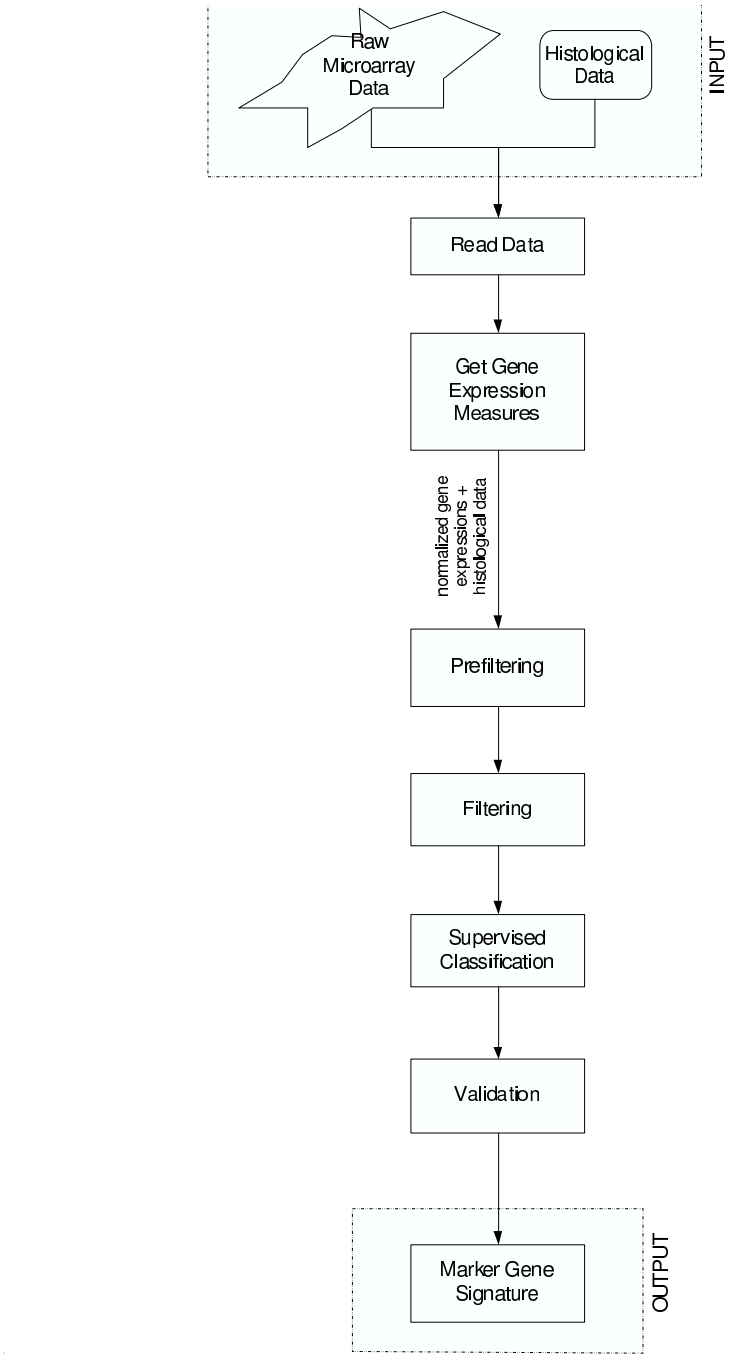


Figure 2.6: Supervised classification design in microarray data analysis.

2.4.2 Development Tools

The development tools are R [43] and Bioconductor [14]. The manifold and the reliability of these tools allow to perform quickly complex analysis design. Moreover these tools are open-source what allows to keep a total control during the development. More precisely, R version 1.9.0 and Bioconductor version 1.4 are used.

2.5 Gene Ontology

The signature of the supervised classification may contain tens or hundreds of genes. The common task is to translate this list of genes into a better understanding of the involved biological phenomena. Currently, this is done through a tedious combination of searches through the literature and a number of public databases. Fortunately useful tools allow to annotate automatically a list of genes.

To obtain some biological information, all genes were annotated according to known function using the Gene Ontology Consortium categories [30]: biological process, cellular component and molecular function. The GO consortium is setting a "dynamic controlled vocabulary that can be applied to all organisms even as knowledge of gene and protein roles in cells is accumulating and changing".

Onto-Express A java-based program called Onto-Express [40] was used to determine whether clusters of genes with similar expression profiles were enriched in specific GO functional categories. Based on the genes present on the chip, Onto-Express calculated the expected number of occurrences of each functional category in each cluster. The probability model best suited to calculate the significance values would use a hypergeometric distribution [47, 41]. For a microarray experiment, when the number of genes on the chips is $N \simeq 10000$ and the number of selected genes is $K \simeq 100 = N/100$, the binomial approximates well the hypergeometric and therefore, the hypergeometric was not implemented. The χ^2 was also proposed for similar problems [37]. For the current experiments, a binomial model was selected to calculate the probability that each functional category was over-represented in a cluster and the p-values were corrected for the multiplicity using the False Discovery Rate method. This method controls the expected number of false rejections among the rejected hypothesis [6]. Onto-Express provided information about the statistical significance of each of the pathways and categories represented by the genes in each cluster. The ontogeny of the biological system can for instance be displayed as a pie with its different parts representing the relative amount of the biological processes affected by the system under study.

Chapter 3

Results

After the hybridization process, about 161 Affymetrix chips are available:

- 77 chips for the JRH population
- 65 chips for the IGR population
- 19 chips from the Karolinska population

The results concerns the quality assessment intra and inter-populations, the classification of the previously selected experiments and the interpretation in using Gene Ontology tools.

3.1 Quality Assessment

Just a part of quality assessment data is shown here because of the huge amount of files generated by this process. The quality is assessed with the probe array image, the average background, the spike controls and RNA degradation, the detection calls, the scaling factor and the box plots for PM intensities.

The source code for quality assessment (*medic_qc.R*) is given in appendix E.

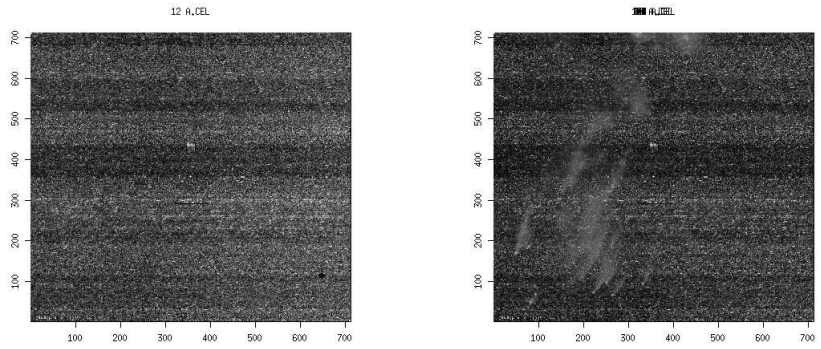
3.1.1 Probe Array Image

The inspection of the probe array images is very subjective. Here are examples of a normal chip and a bad chip (see figure 3.1). This quality criterion can be assessed one chip by one. Only one chip hgu133a in the JRH population seems to be bad (artifact like a "wave" on the chip, see table 2.1).

3.1.2 Average Background

Since the average background values should be similar among all the experiments, all the values have to be drawn in a same plot (see figure 3.2). An additional population is taken into account: we have received 68 *CEL* files from the Karolinska Institute. These *CEL* files come from the same group of tumors than the 19 *CEL* files from Karolinska which have been hybridized at the IJB[1]. In such plot, four populations are represented and separated by vertical lines: JRH, IGR, Karolinska19 and Karolinska68.

We can notice that a large part of the experiments have an average background higher than the recommended value of 100 (see table 2.1). Moreover there is a significant difference between populations. A permutation test can assess this difference: the R function *perm.test* [55] permits



(a) (b)

Figure 3.1: Probe array images. (a) Normal chip from chip hgu133a of the JRH population. (b) Bad chip from chip hgu133a of the JRH population (artifact like a "wave" on the chip).

Chip hgu133a	
Populations	**p-value**
JRH <-> IGR	1.903e-9
JRH <-> Karolinska19	1.08e-10
IGR <-> Karolinska19	0.06725

Chip hgu133b	
Populations	**p-value**
JRH <-> IGR	2.661e-4
JRH <-> Karolinska19	0.03496
IGR <-> Karolinska19	0.6224

Table 3.1: Permutation tests between each pair of population to assess the average background difference for the chip hgu133a and hgu133b. If the p-value is small, the null hypothesis is rejected, i.e. the populations are significantly different.

to perform a permutation test between two samples of different size. Results of such a test is given in table 3.1.

Surprisingly these differences are less significant for the chip hgu133b. None valid explanation has been found yet.

3.1.3 Spike Controls and RNA Degradation

These quality criteria can be assessed in each population separately. We can see in the figure 3.3 the results for the IGR population: only the beta actin 3'/5' ratio is higher than the limit fixed by Affymetrix (see table 2.1) in some experiments. However, Affymetrix considers that one ratio can be higher without affecting significantly the quality of the chip. All the spike controls are detected as Present in compliance with Affymetrix guidelines.

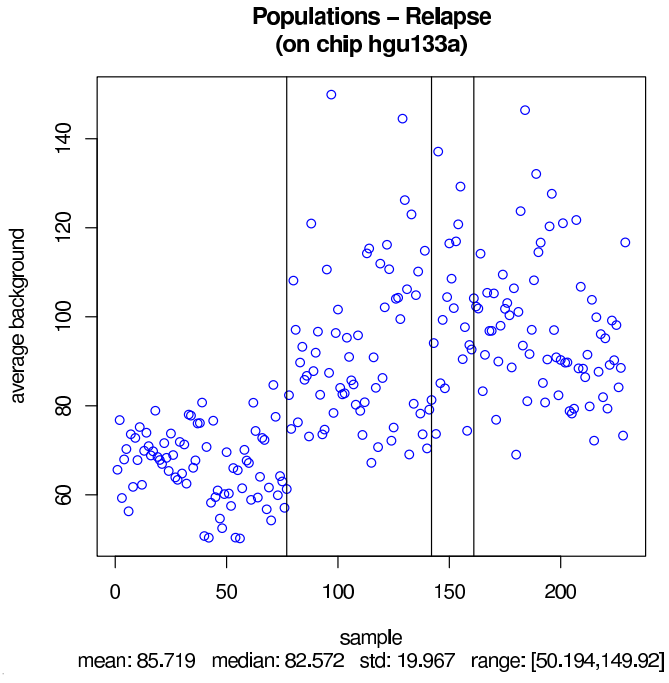
Similar observations can be done for the other populations (not shown here).

3.1.4 Detection Calls

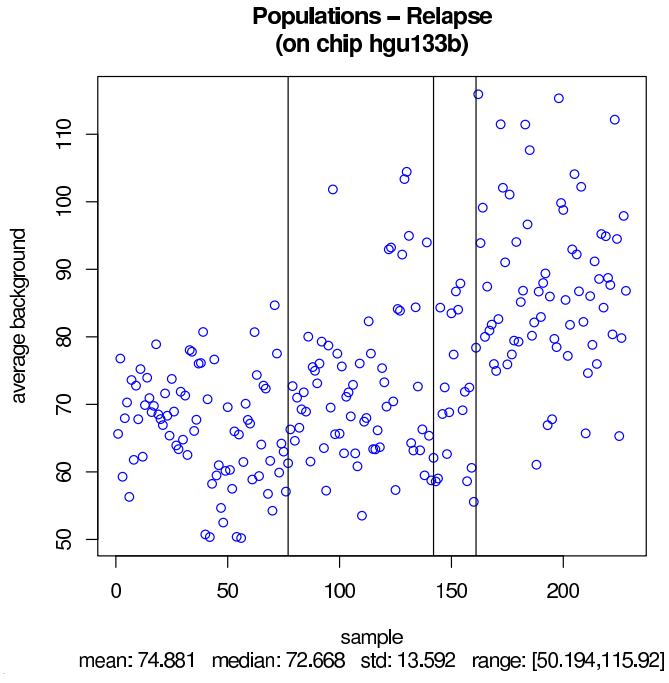
These quality criteria can be assessed in each population separately. We can see that the percentage of present (figure 3.4) and absent (figure 3.5) detection call are about 50% in compliance with Affymetrix guidelines (see table 2.1). Moreover, the percentage of marginal detection call is very low (see figure 3.6).

Similar observations can be done for the chip hgu133b and for the other populations (results not shown here).

[1]There are *CEL* files coming from the same biological samples between the 19 of the IJB and the 68 from Karolinska. An interesting analysis of reproducibility between laboratories can be done.

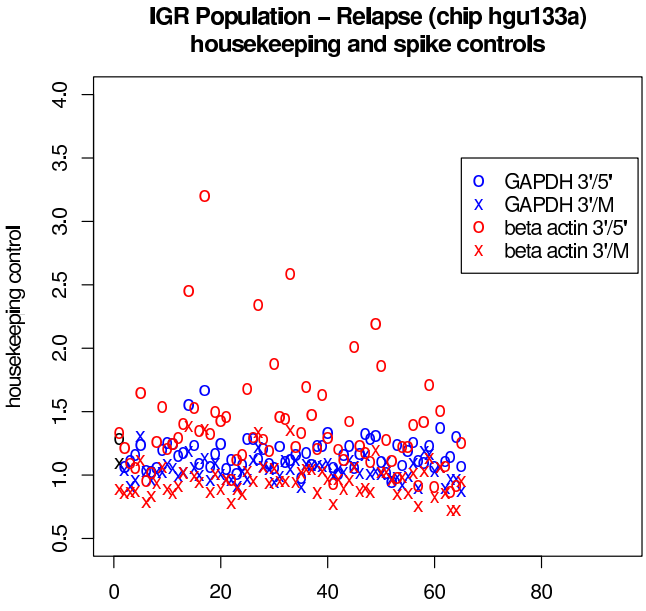


(a)

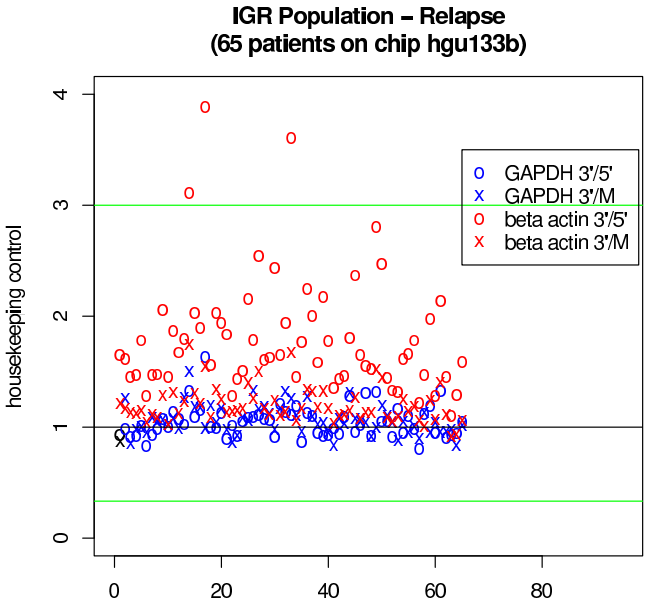


(b)

Figure 3.2: Average background values for the JRH, IGR and the Karolinska populations. The y-axis represents the average background and the x-axis represents the identifiers for each experiment. The three vertical lines separate the different populations: JRH | IGR | Karolinska19 | Karolinska68. (a) Chip hgu133a. (b) Chip hgu133b.



(a)



(b)

Figure 3.3: Spike controls and internal control genes for the IGR population. The y-axis represents ratio for the internal control genes and the x-axis represents the identifiers for each experiment. The green horizontal lines represent the ratio interval recommended by Affymetrix guidelines (ratio belongs to [0.33, 3]). (a) Chip hgu133a. (b) Chip hgu133b.

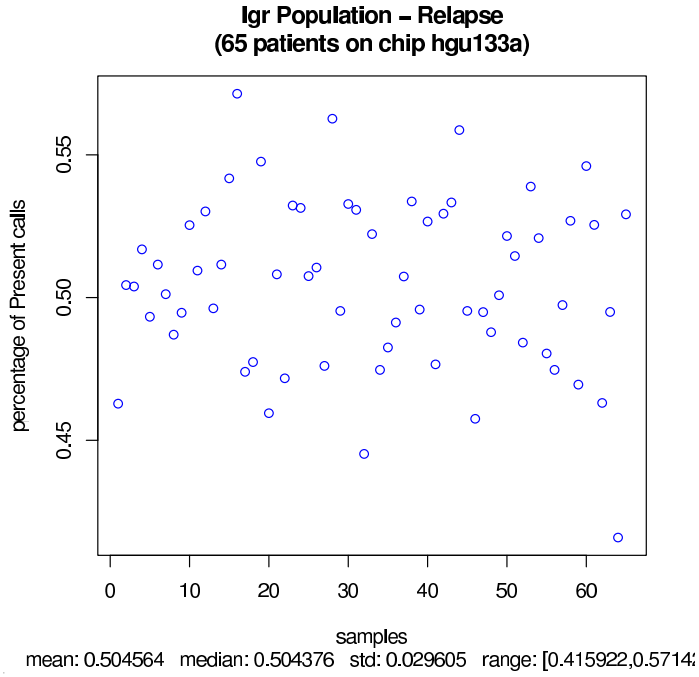


Figure 3.4: Detection calls. Percentage of present calls for the chip hgu133a in the IGR population. The y-axis represents the percentage of present calls and the x-axis represents the identifiers for each experiments.

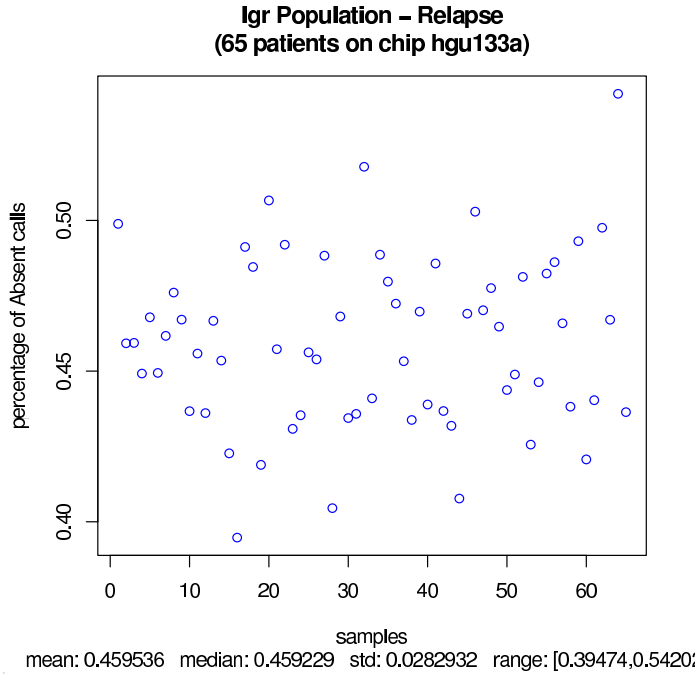


Figure 3.5: Detection calls. Percentage of absent calls for the chip hgu133a in the IGR population. The y-axis represents the percentage of marginal calls and the x-axis represents the identifiers for each experiments.

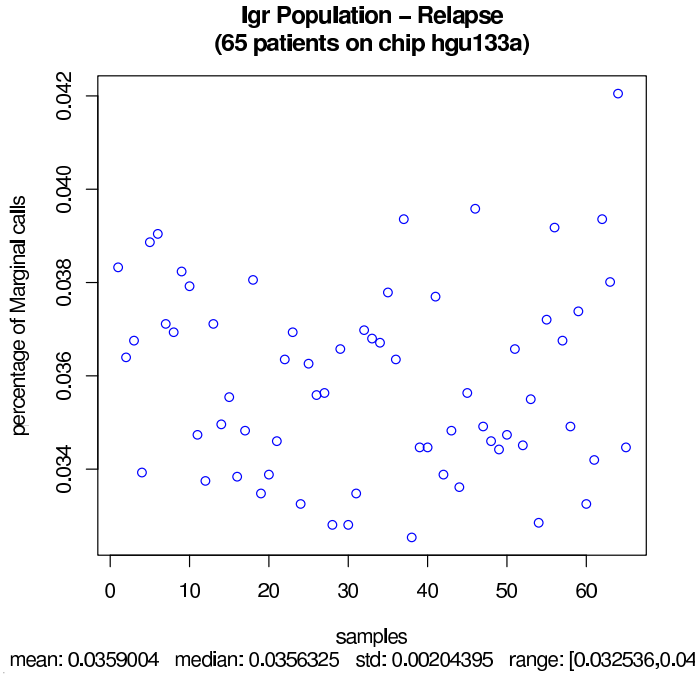


Figure 3.6: Detection calls. Percentage of marginal calls for the chip hgu133a in the IGR population. The y-axis represents the percentage of absent calls and the x-axis represents the identifiers for each experiments.

3.1.5 Scaling Factor

The scaling factors are compared among all the populations. In an ideal case, all the scaling factors belong to an interval of a three-fold change around a specific value. In the figure 3.7, this specific value is the median[2] and the area between the two green horizontal lines represent the interval of a three-fold change[3]. All the experiments which do not belong to this interval are marked in red, the others are marked in blue. According to this criterion, all the "red" experiments have to be discarded for the analysis (see table 2.1).

We can see significant differences between populations for the chip hgu133a. A permutation test can assess this difference: the R function *perm.test* [55] permits to perform a permutation test between two samples of different size. Results of such a test is given in table 3.2.

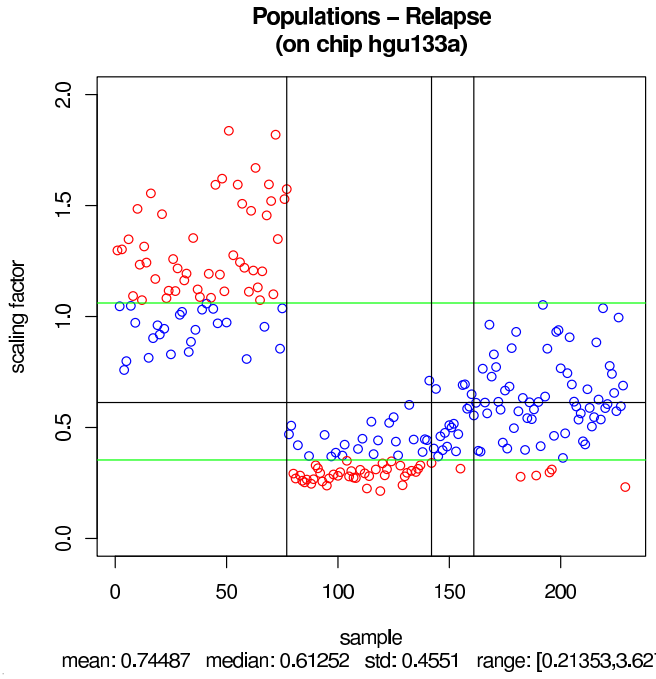
Chip hgu133a	
Populations	**p-value**
JRH <-> IGR	1.166e-6
JRH <-> Karolinska19	0.1066
IGR <-> Karolinska19	5.118e-7

Chip hgu133b	
Populations	**p-value**
JRH <-> IGR	3.74e-7
JRH <-> Karolinska19	0.4476
IGR <-> Karolinska19	0.002979

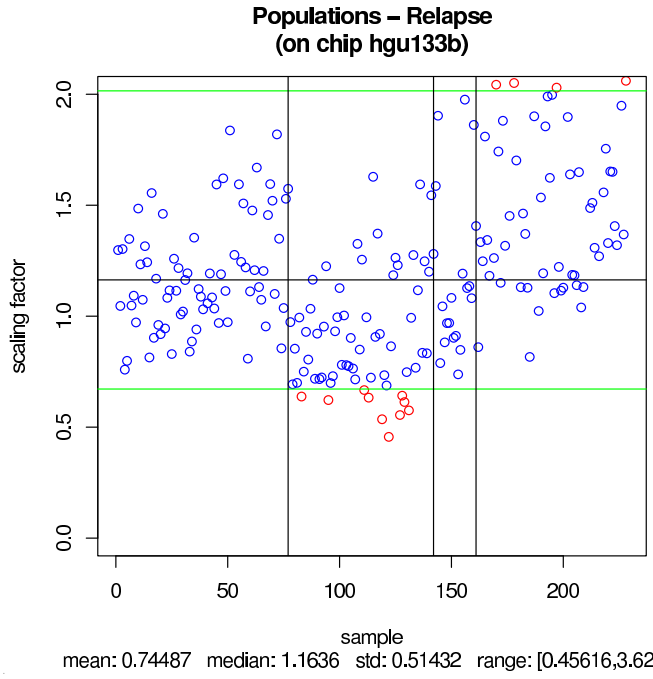
Table 3.2: Permutation tests between each pair of population to assess the scaling factor difference for the chip hgu133a and hgu133b. If the p-value is small, the null hypothesis is rejected, i.e. the populations are significantly different.

Surprisingly these differences are less significant for the chip hgu133b. None valid explanation has been found yet.

[2] It is an arbitrary choice: the mean or another value can be selected.
[3] If the median equals to 1, the interval of three-fold change is [0.33, 3].



(a)



(b)

Figure 3.7: Scaling factors for the JRH, IGR and the Karolinska populations. The y-axis represents the scaling factors and the x-axis represents the identifiers for each experiment. The three vertical lines separate the different populations: JRH | IGR | Karolinska19 | Karolinska68. The black horizontal line is the median of all scaling factors. The two green lines represent the recommended maximum factor between all the scaling factors. (a) Chip hgu133a. (b) Chip hgu133b.

3.1.6 Box Plots for the PM Intensities

Box plots for all the populations can be computed to compare the median and the interquartile interval of each experiment (see table 2.1 for details). Unfortunately, a common workstation is insufficient to compute such number of patients[4]. Each population has been taken separately to compute the box plots for the PM intensities (see figure 3.8, 3.9, and 3.10 for the box plots of raw data, data after MAS normalization and after RMA normalization respectively).

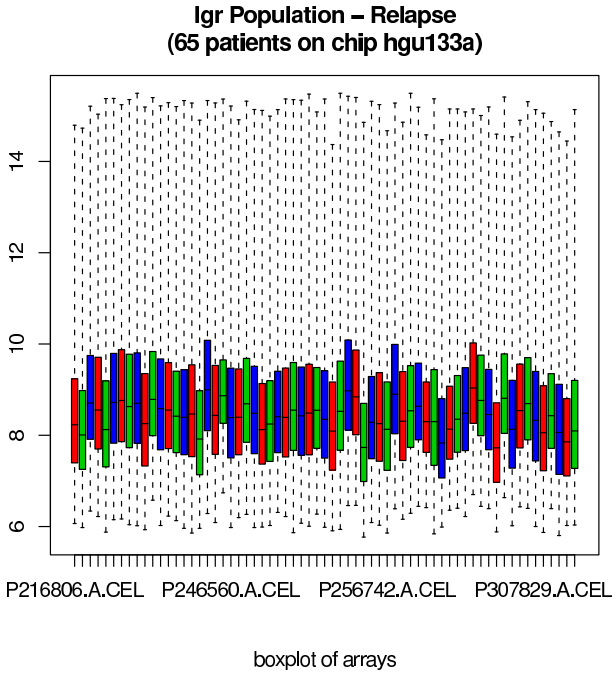


Figure 3.8: Box plots for the raw data from the IGR population (chip hgu133a). The y-axis represents the boxes from log of raw gene expressions and the x-axis represents the identifiers for each experiment (only few of them are displayed). For details, see the function *boxplot* from the *affy* package [8].

3.1.7 Preliminary Conclusion

The probe array image allows us to detect one bad chip. The detection calls, spike controls and RNA degradation have not detected poor quality experiments. The average background and the scaling factor criteria tend to show that it exists a significant difference between the populations. The different experiments are not necessary comparable and a data transformation have to be done before the analysis (not yet investigated).

3.2 Classification

Preliminary results will be shown here. When the classification has been implemented, only 99 patients were taken into account. Actually only 47 patients from JRH and 52 patients from IGR were valid in terms of quality and histological data (the 19 samples and the 68 *CEL* files from Karolinska were not available). For these 99 patients, the chip hgu133a and hgu133b are used.

The results are represented by four main sections:

[4]Actually, Bioconductor functions have to be re-implemented to support larger amount of data.

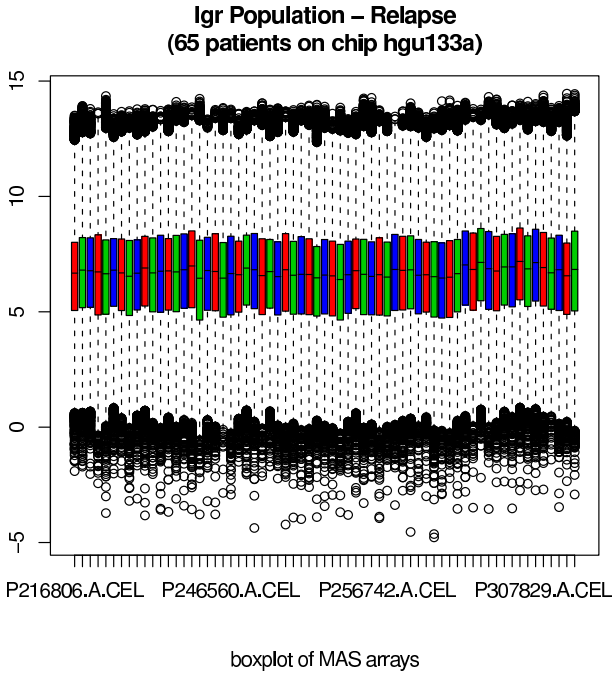


Figure 3.9: Box plots for the MAS normalized data from the IGR population (chip hgu133a). The y-axis represents the boxes from log of MAS normalized gene expressions and the x-axis represents the identifiers for each experiment (only few of them are displayed). For details, see the function *boxplot* from the *affy* package [8].

1. Selection of sub-optimal parameters by structural identification
2. Misclassification rate during the feature selection
3. Marker genes
4. Misclassification rate

The figure 3.11 illustrates the implemented classification design.

The RMA set of analysis functions is used to carry out the microarray data preprocessing (see appendix B). The prefiltering and filtering steps are not be carried in this classification design (the forward feature selection on the ranked genes replace these two steps).

All the results concern the KNN [46, 59] classifier[5].

3.2.1 Structural Identification

The structural identification is performed with all the patients (see appendix B.5.1). The number of neighbours $k \in [3, 4, 5, ..., 30]$ is tested to find a sub-optimal value. We can see the evolution of the global misclassification rate[6] in figure 3.12.

Only the global misclassification rate is used to assess the quality of the classifier[7]. This may cause a problem because the population is not balanced between the two classes.

The code of *medic_ struct_ id.R* is given in appendix I.

[5] Results concerning the SVM [12] classifier can be seen in appendix C.

[6] The global misclassification rate is given by $MISCLASSIF = (FP + FN)/(FP + TP + FN + TN)$.

[7] Actually the function *tune.foo* in R does not allow to chose an assessment function and weights for the different classes.

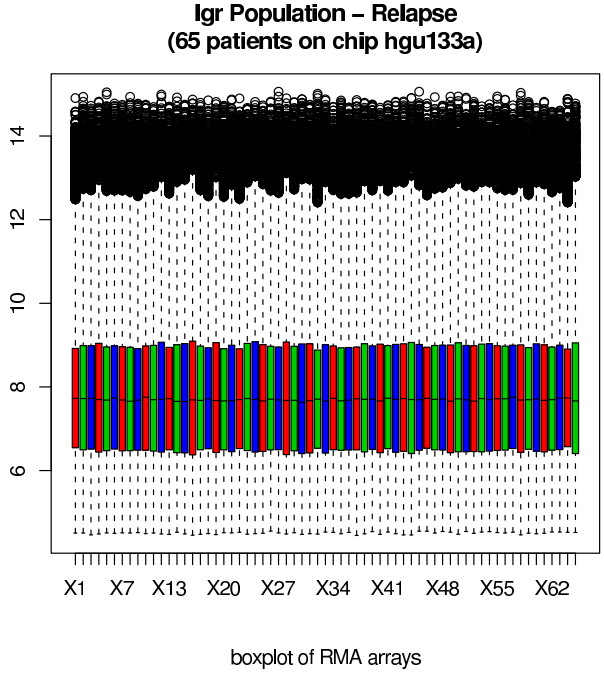


Figure 3.10: Box plots for the RMA normalized data from the IGR population (chip hgu133a). The y-axis represents the boxes from log of RMA normalized gene expressions and the x-axis represents the identifiers for each experiment (only few of them are displayed). For details, see the function *boxplot* from the *affy* package [8].

3.2.2 Misclassification Rate during Feature Selection

The feature selection is described by the algorithm 2 and illustrated in figure 3.11.

At each global leave-one-out [54] step (see figure 3.11 and algorithm 3 for the pseudo code of the whole classification), a feature selection is performed. We can see the evolution of the misclassification rate in the figure 3.13. On each figure, we can see a very high misclassification rate for the false negatives on the contrary of the false positives. It is a preliminary indication that the classifier is poor on our training set.

3.2.3 Marker Genes during Feature Selection

As described in the section 3.2.2, the feature selection is performed at each leave-one-out step to select a set of marker genes. It would interesting to see if the set of selected genes is robust against the training set. To assess the robustness, the number of occurrences of each gene selected during the leave-one-out is computed (see figure 3.14). In an ideal case, the set of marker genes is composed of genes which appear 99 times during the leave-one out. This would mean that the set of marker genes is always the same but it is not the case here: the majority of marker genes are selected only one times but some genes are selected more than once (maximum 6 times). We see that the marker genes are very dependent on the training set.

3.2.4 Misclassification Rate

The misclassification rate at each global leave-one-out step[8] can be represented by the three histograms on figure 3.15. We can see the good classification for the non-relapse patients and the

[8]For each step of the global L-O-O, the misclassification rate is estimated by a L-O-O during feature selection (see figure 3.11). We have 99 estimations of misclassification rate, represented on the histograms in figure 3.15.

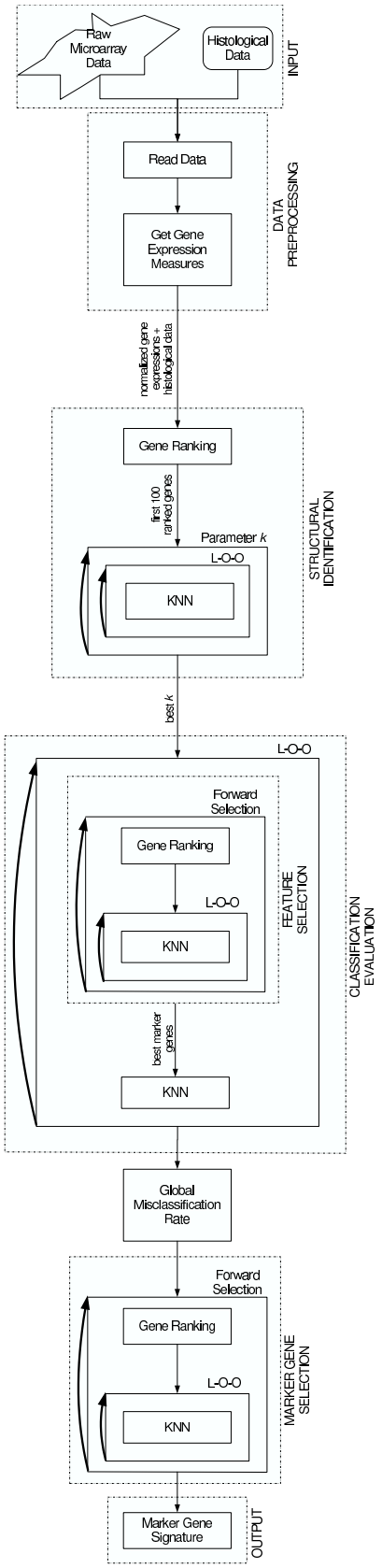


Figure 3.11: KNN classification design.

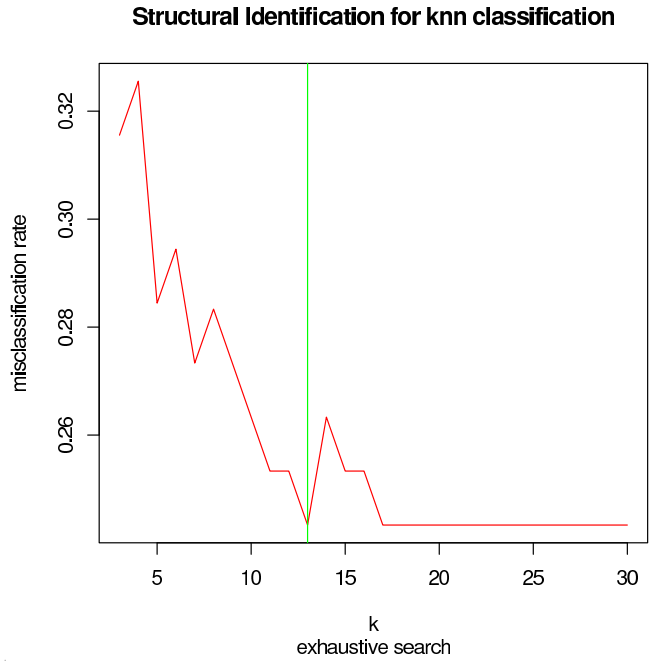


Figure 3.12: Structural identification for the KNN classifier. The sub-optimal number of neighbours equals to 13.

dramatically poor performance for the relapse patients.

The misclassification rate computed by a global leave-one-out on all the patients with the maker gene signature, corresponds to 4/75 false positives and 21/24 false negatives. The trend is to classify patients in the non-relapse class every time.

The results for the classification with SVM are similar but the trend is the inverse: patients will be classify in the relapse class every time (justification is given in appendix C).

3.2.5 Signature of Marker Genes

After the performance evaluation of the classification technique by leave-one-out, a signature of marker genes can be computed from all the patients. The feature selection is carried out to determine the marker genes (see figure 3.16).

3.2.6 Preliminary Conclusion

The implemented analysis design tries to avoid overfitting although some step (e.g. structural identification, see appendix B.5.1 and B.5.2) have been simplified because of computer resources. The quality criterion of classification (see appendix B.5.3) is so strict with the FN that the classifier is poor in global misclassification rate. These two facts can explain the poor performance of the classifier.

3.3 Gene Ontology

The Gene Ontology can be interrogated to obtain biological information about the two marker genes but only the probe set id 223529_s_at exists in the GO:

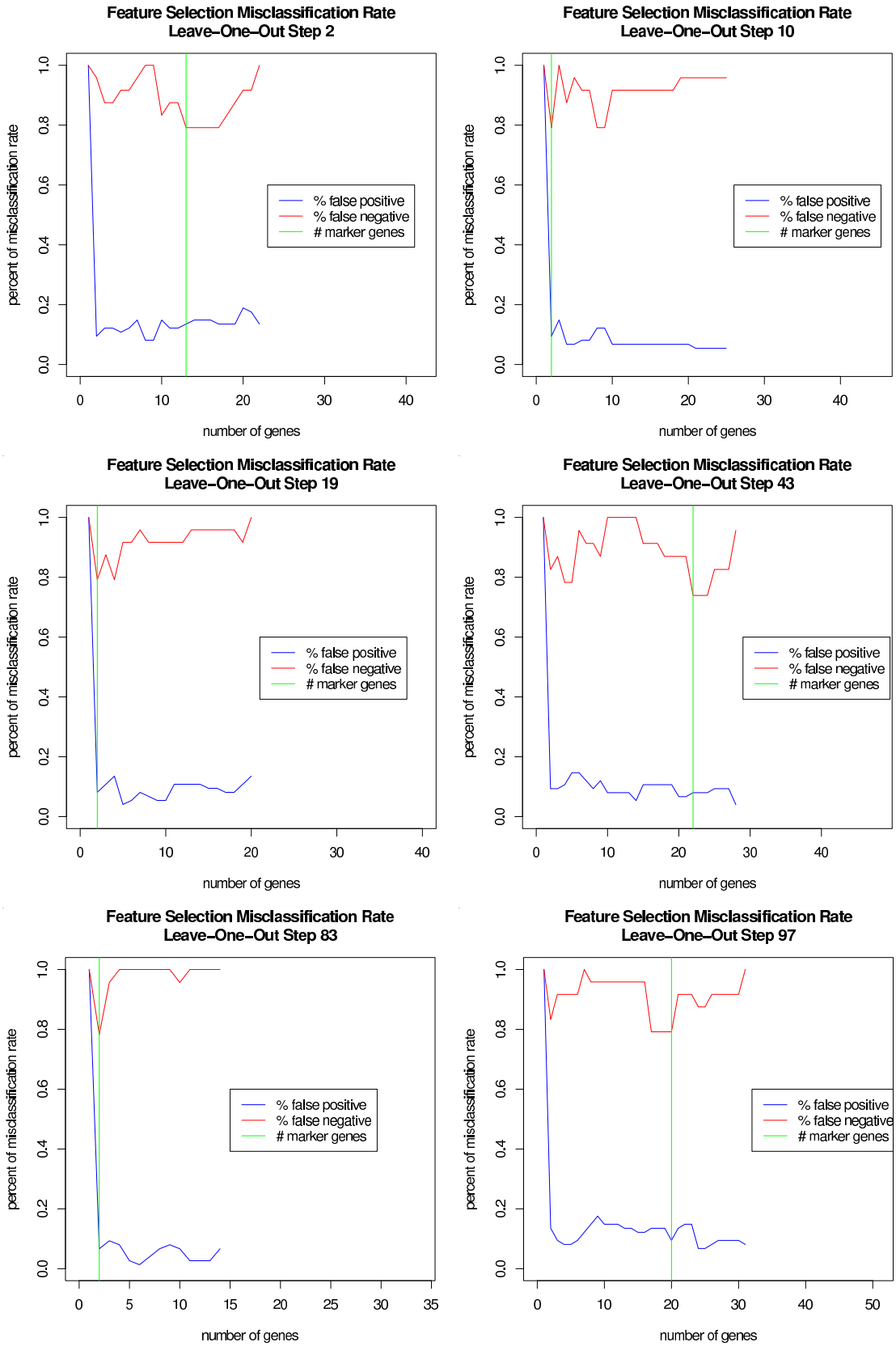


Figure 3.13: Subset of figures which represent the evolution of misclassification rate during the feature selection in KNN classifier. There are 99 figures but only few of them are shown here.

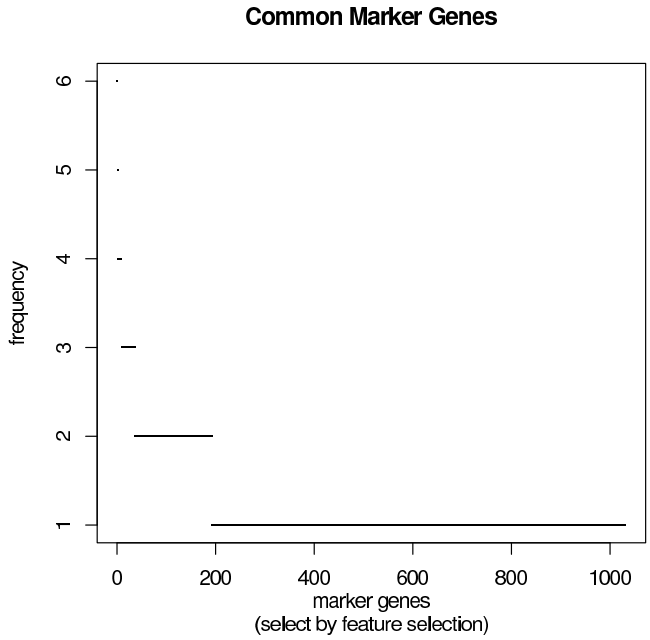


Figure 3.14: Marker genes selected during the feature selection for the KNN classifier.

Biological process

GO ID	Function Name	Probe	Gene Symbol	Unigene Cluster	LocusLink ID
GO:0009116	nucleoside metabolism	224529_s_at	NT5C1A	307006	84618

Cellular component

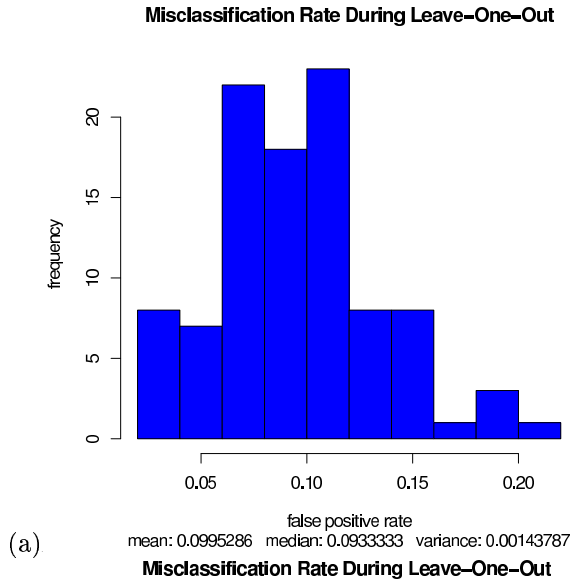
GO ID	Function Name	Probe	Gene Symbol	Unigene Cluster	LocusLink ID
GO:0005829	cytosol	224529_s_at	NT5C1A	307006	84618

Molecular function

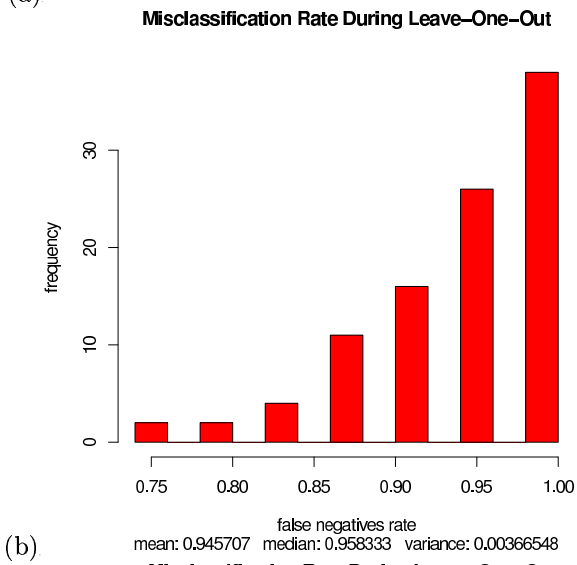
GO ID	Function Name	Probe	Gene Symbol	Unigene Cluster	LocusLink ID
GO:0008253	5'-nucleotidase activity	224529_s_at	NT5C1A	307006	84618

Such a gene is not known to be involved in breast cancer (see [25] to have a good overview of known genes involved in breast cancer).

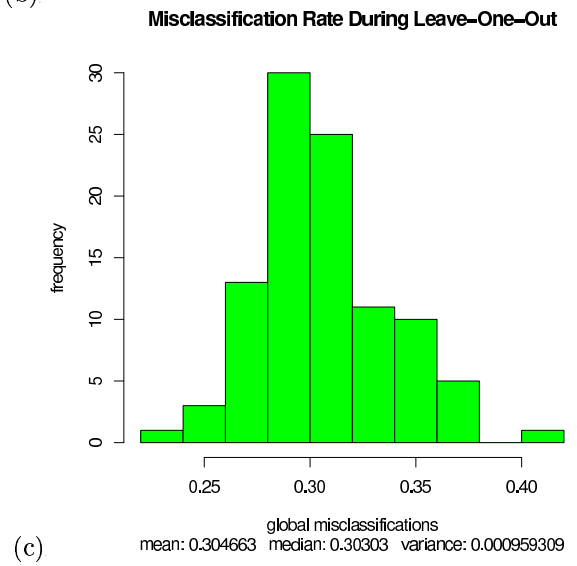
Another example of the use of GO to interpret marker gene signatures is given in appendix D.



(a)

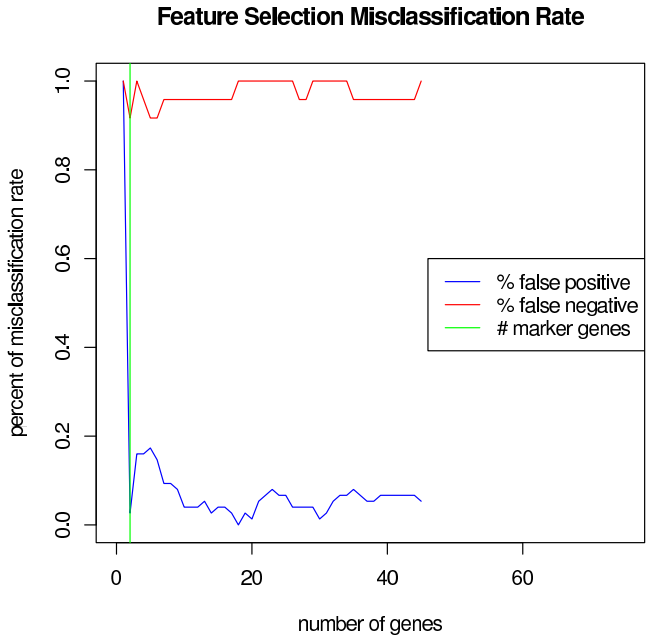


(b)



(c)

Figure 3.15: (a) Misclassification rate for the false positives. (b) Misclassification rate for the false negatives. (c) Global misclassification rate.



2 marker genes: **224529_s_at** and **223176_at**

Figure 3.16: Feature selection for the KNN classifier to determine the signature of marker genes.

Chapter 4

Discussion

Although microarray analysis of breast cancer has provided valuable information for classifying tumors on a molecular basis, in predicting the clinical outcome, there are several significant issues that need to be addressed before this powerful tool can be brought into the clinic. Within the next decade, as the cost of conducting microarray experiments is expected to decrease, more academic investigators will include this technology in their arsenal of tools. Microarrays might then be routinely used for diagnosis and for monitoring desired and adverse outcomes of therapeutic interventions. Nevertheless, despite the potentially enormous benefits of microarrays to public health, challenges must be met to ensure the seamless incorporation of this technology into medical practice. Quality control and assurance must be established; some guidelines exist already but standards have to be set among the different microarray platforms.

The determination of appropriate levels of analytical and biological validation needed for each medical application of microarrays and their supporting computer based bioinformatics systems raises new challenges. These needs are exemplified by a comparison of results generated from experiments done on different microarray platforms (cDNA versus short-oligonucleotides) showed that, although there was a similar pattern of expression for some of the genes, there was a large variation in expression between the tested platforms [21, 56]. All this confirms the requirement for standardization of the microarray technique as well as the need to validate the expression pattern of genes of interest by an alternative RNA quantitative method, such as Northern Blot, quantitative RT-PCR or RNase protection assay, above all when precise quantification is mandatory. Such multi-platform validation study is currently under investigation at the IJB. Indeed we have access to several biological samples hybridized with two different microarray platforms (Affymetrix/Agilent) and different populations.

The number of relevant publications in the field of microarrays is increasing exponentially: during the years 1995-1997, there were fewer than ten reports featuring microarray data; by the time that this review was written, however, approximately 5000 reports had been published in this burgeoning field, with 240 in the breast cancer research area. Thus, in order to be able to ensure the interpretability of the experimental results generated by the use of microarrays as well as their potential independent validation, there is a crucial need for standardization of data collection. Unfortunately, this is currently lacking in the majority of the published microarray reports. Such an initiative has been proposed by the Microarray Gene Expression Data Society (MGED), which developed guidelines for submitting microarray data for publication known as "Minimum Information About a Microarray Experiment" (MIAME [3]). These guidelines, which intend to facilitate the interpretation and verification of microarray results, are currently accepted by most journals and must be applied by every investigator involved in microarray studies [50].

An important challenge posed by this technology lies in discerning the biological meaning of the huge volume of array data. Although no exact rule exists regarding the statistical approaches required for such analysis, various methods are continually being developed. Because DNA microarrays are used for a variety of different purposes (i.e. "class comparison", "class prediction" or "class discovery" studies), the analysis strategy should be determined in light of the overall

objectives of the study. Moreover, because it is likely that gene expression profiles will provide information that might affect clinical decision making, such profiling studies must be performed with statistical rigor, and must be reported clearly with unbiased statistics, which again are lacking in several microarray studies published so far [49].

4.1 Future Works

This preliminary study has highlighted some problems that require future works:

- Parallelism: the requirements in computer resources are very high in this research field. The huge amount of microarray data and the complexity of the classification design imply efficient parallelized implementations. R functions have been originally implemented for small experiment with a reasonable amount of data. Such functions could be reimplemented to perform complex analysis design with large amount of data in using cluster parallelization.
- Statistical framework for quality assessment: the need of fixed quality standard implies also the need of a statistical framework to assess such quality criteria. Permutation test have been used to compare populations (see section 3.1.2 and 3.1.5) but more complex statistical tests could be performed.
- Preprocessing data: the choice of normalization (e.g. MAS or RMA) and the prefiltering methods influence considerably the results and the interpretability of microarray analysis. This influence is not well quantified and further works can be done in this field [9].
- Criterion for misclassification rate: populations of patients coming from clinical studies are often unbalanced (different number of patients for class 0 and 1). The choice of a criterion for misclassification rate is not trivial [48] (e.g. global misclassification rate, $FP/(FP+TP)$ or $FN/(FN+TN)$) because an error type is often more important than others (FN in the case of breast cancer classification, see section B.5.3). Such a choice can influence the performance of classifiers.
- Marker gene stability: we have seen in section 3.2.3 that the marker genes selected during the feature selection are very dependent on the training set. Techniques centered of robustness could be studied to avoid this problem.
- Feature selection: the implemented feature selection, i.e. the forward selection, is not necessary the most efficient method although simple. Currently, more complex feature selection methods have been studied [20, 33] and could be implemented in this research field.
- Independent validation set: we could use another independent population to validate our marker gene signature (e.g. Karolinska68).
- Signature validation and refinement: we have already seen that the marker genes can be easily annotated in using recent GO tools (see Onto-Express in section 2.5 and appendix D for an example). However an automatic search in breast cancer literature could be useful to complete such annotations.
- Multi-laboratory variability: as mentioned in section 3.1.2 (footnote), we have several biological samples hybridized in two different laboratories using the same microarray platform. An analysis can be performed to assess the laboratory variability.
- Multi-platform comparison: in a not too distant future, we will have same biological samples hybridized in two different microarray platforms (Agilent and Affymetrix). The characteristics of each platform could be studied to improve the potential of meta-analysis.

Bibliography

[1] R.D. Gelber A.S. Coates-B. Thurlimann A. Goldhirsch, W.C. Wood and H.J. Senn. Meeting highlights: Updated international expert consensus on the primary therapy of early breast cancer. *J. Clin. Oncol.*, 21(17):3357–3365, 2003.

[2] R.D. Gelber A. Goldhirsh, J.H. Glick and H. Senn. Meeting highlights: International consensus panel on the treatment of primary breast cancer. *Journal of National Cancer Institute*, 90(1601-1608), 1998.

[3] Ball CA Causton HC-Gaasterland T Glenisson P Holstege FC Kim IF Markowitz V Matese JC Parkinson H Robinson A Sarkans U-Schulze-Kremer S-Stewart J Taylor R Vilo J Aach J, Ansorge W and Vingron M. Minimum information about a microarray experiment (miame)-toward standards for microarray data. *Nature Gen*, 29(4):365–371, 2001.

[4] Affymetrix. Affymetrix products. http://www.affymetrix.com/.

[5] Affymetrix. *GeneChip Expression Analysis*, 2002.

[6] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J R Stat oc*, 57:289, 1995.

[7] M. Astrand B.M. Boldstad, R.A. Irizarry and T.P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on bias and variance. *Bioinformatics*, 19(2):185–193, 2003.

[8] B. Bolstad. Affy: Built-in processing methods. Technical report, Bioconductor, 2004.

[9] Astrand M Bolstad BM, Irizarry RA and Speed TP. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–193, 2003.

[10] breastcancer.org team. breastcancer.org: a non profit organization for breast cancer education. http://www.breastcancer.org.

[11] V. Durbecq L. Dal Lago M. Lacroix F. Cardoso C. Sotiriou, C. Desmedt and M. Piccart. Molecular oncology of breast cancer. Chapter 6: Genomic and Molecular Classification of Breast Cancer (in press), 2004.

[12] C. Chang and H. Lin. A library for support vector machines. http://www.csie.ntu.edu.tw/cijlin/libsvm.

[13] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[14] V. Carey M. Dettling S. Dudoit B. Ellis L. Gautier R. Gentleman J. Gentry K. Hornik T.Hothorn W. Huber S. Iacus F Leisch-J. MacDonald-M. Maechler C. Smith G. Smyth A. Rossini F. Hutchinson G. Sawitzki L. Tierney J.Y.H. Yang J. Zhang D. Bates, B. Bolstad. Bioconductor.

[15] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification.* John Wiley and sons, 2001.

[16] P. Eifel et al. National institutes of health consensus development conference statement: Adjuvant therapy for breast cancer. *Journal of National Cancer Institue*, 93(979-989), 2001.

[17] Early Breast Cancer Trialists' Collaborative Group. Polychemotherapy for early breast cancer: an overview of the randomized trials. *Lancet*, 352:930–942, 1998.

[18] Panavally S. Saal L.H. Borg A. Ferno M. Peterson C. Gruvberger S., Ringner M. Chen Y. and Meltzer P.S. Estrogen receptor status in breast cancer is associated with remarkably distainct gene expression patterns. *Cancer Res*, 61(16):5979–5984, 2001.

[19] Francois C Yasmin D Beazer-Barclay Kristen J Antonellis Uwe S Irizarry RA, Bridget H and Speed TP. Exploration, normalization, and summarization of hight density oligonucletotide array probe level data. *Bioinformatics*, 2003. in press.

[20] R. Kohavi and G. H. John. Wrappers for feature subset selection. *AIJ*, 97(1-2):273–324, 1997.

[21] Butte AJ Ohno-Machado L Kuo WP, Jenssen TK and Kohane IS. Analysis of matched mrna measurements from two different microarray technologies. *Bioinformatics*, 18(3):405–412, 2002.

[22] B. Haibe-Kains V. Bours J. Boniver L. de Leval, C. Herens and C. Sotiriou. Gene expression profiling of diffuse large b-cell lymphoma: correlation with bcl-2 and bcl-6 rearrangements. submitted, 2004.

[23] L. Cope L. Gautier, R. Irizarry and B. Boldstad. Description of affy. Technical report, Bioconductor, 2004.

[24] W.M. Keck Foundation: Biotechnology Resource Laboratory. Affymetrix genechip technology overview. http://keck.med.yale.edu/affymetrix/technology.htm, 2002.

[25] M. Lacroix and G. Leclerq. Running head: Comparison of cell lines and tumours. Relevance of breast cancer cell lines as models for breast tumours.

[26] M.J. van de Vijver Y.D. He A.A.M. Hart M. Mao H.L. Peterse K. van der Kooy M.J. Marton A.T.Witteveen G.J. Schreiber R.M.Kerkhiven-C. Roberts-P.S. Linsley R. Bernards L.J. van't Veer, H. Dai and S.H. Friend. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415, 2002.

[27] Berk A Krieger M Matsudaira P Lodish H, Kaiser CA and Scott MP. *Molecular Cell Biology*. Sara Tenney, 4 edition, 2003.

[28] Deloreny M. Swiss instittute of bioinformatics. http://www.isrec.ch.

[29] Lacroix M and Leclercq G. Relevance of breast cancer cell lines as models for breast tumors: an update. *Breast Cancer Res and Treat*, 415:530–536, 2004.

[30] JA. Blake D. Botstein H. Butler JM. Cherry AP. Davis K. Dolinski SS. Dwoght JT. Eppig MA. Harris DP. Hill L. Issel-Tarver-A. Kasarskis-S. Lewis JC. Matese JE. Richardson M. Ringwald GM. Rubin G. Sherlock M. Ashburner, CA. Ball. Gene ontology: tool for the unfication of biology. the gene ontology consortium. *Nat Genet*, 25:25–29, 2000.

[31] JF. Laes B. Hennuy F. Lallemand I. Gonze F. Cardoso M. Piccart G. Leclercq M. Lacroix, B. Haibe-Kains and C. Sotiriou. Gene regulation by phorbol 12-myristate 13-acetate (pma) in two jighly different breast cancer cell lines. *Oncology Report*, 2004.

[32] H. Dressman E. Huang S. Ishida R. Spang H. Zuzan J.A. Olson J.R. Marks M. West, C. Blanchette and J.R. Nevins. Predicting the clinical status of human breast cancer by using gene expression profiles. *PNAS*, 98(20):11462–11467, 2001.

[33] Oden Maron and Andrew W. Moore. The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, 11(1-5):193–225, 1997.

[34] D. Meyer. *Support Vector Machines: The interface to libsvm in Package e1071.* Techische Universitat Wien, Austria, 2004.

[35] C.J. Miller. Description of simpleaffy: Easy analysis routines for affymetrix data. Technical report, The Paterson Institute Bioinformatics Group, 2004.

[36] L. van't Veer H. Dai A.A.M. Hart D.W. Voskuil G.J. Schreiber J.L. Peterse C. Roberts M.J. Marton M. Parrish D. Atsma-A. Witteveen-A. Glas L. Delahaye T. van der Velde H. Bartelink S. Rodenhuis E.T. Rutgers S.H. Friend M.J. van de Vijver, Y.D. He and R. Bernards. A gene expression signature as a predictor of survival in breast cancer. *The new England*, 347(25):1999–2009, 2002.

[37] Z. Wang MZ. Man and Y. Wang. Powersage: Comparing statistical tests for sage experiments. *Bioinformatics*, 16:953–959, 2000.

[38] B. Samans O. Hartmann and H. Schafer. Low level analysis for affymetrix genechips: Normalization and quality control. Technical report, Insitiute of Medical Biometry and Epidemiology. Faculty of Medicine and Hospital, Philippe-University, 2003.

[39] J. Costa J. Crowley W.J. Curran A. Deshler S. Fulton C.B. Hendricks M. Kemeny A.B. Kornblith T.A. Louis M. Markman R. Mayer D. Roter P. Eifel, J.A. Axelson. National institutes of health consensus development conference statement: Adjuvant therapy for breast cancer. *J. Natl Cancer Inst.*, 93(13):979–989, 2001.

[40] GC Ostermeier P. Khatri, S. Draghici and SA. Krawetz. Profiling gene expression using onto-express. *Genomics*, 79:266–270, 2003.

[41] RP. Martins GC Ostermeier P. Khatri, S. Draghici and SA. Krawetz. Global functional profiling of gene xpression. *enomics*, 81:98–104, 2003.

[42] Eisen M.B. van de Rijn M. Jeffrey S.S. Rees C.A. Pollack J.R. Ross D.T. Jonhsen H. Aklslen L.A. Fluge O. Pergamenschikov A; Williams C. Zhu S.X Loning P.E. Borresen-Dale A.L. Brown P.O. Perou C.M., Sorlie T. and Botstein D. Molecular portraits of human breast tumours. *Nature*, 406(6797):747–752, 2000.

[43] R Development Core Team. *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria, 2003. ISBN 3-900051-00-3.

[44] F. Collin L.M. Cope B. Hobbs R.A. Irizarrry, B.M. Boldstad and T.R. Speed. Summaries of affymetrix genechip probe level data. *Nucleic Acids Research*, 31(4), 2003.

[45] F. Collin Y.D. Beazer-Barclay R.J. Antonellis U. Scherf R.A. Irizarry, B. Hobbs and T.P. Speed. Normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, 4(2):249–264, 2003.

[46] B.D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996. ISBN 0 521 46086 7.

[47] P. Bhavsar A. Shah-SA. Krawetz S. Draghici, P. Khatri and MA. Tainsky. Onto-tools, the toolkit of the modern biologist: Pnto-express, onto-compare, onto-design and onto-translate. *Nucleic Acids Research*, 31(13):3775–3781, 2003.

[48] Latinne P Saerens M and Decaestecker C. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Comp.*, 14:21–41, 2002.

[49] Dobbin K Simon R, Radmacher MD and McShane LM. Pitfalls in the use of dna microarray data for diagnostic and prognostic classification. *J. Natl Cancer Inst*, 95(1):14–18, 2003.

[50] Microarray Gene Expression Data Society. Minimum information about a microarray experiment (miame). http://www.mged.org/miame, 1999.

[51] Parker J. Hastie T.-Marron J.S. Nobel A. Deng S. Johnsen H. Pesich R. Geister S. Demeter J. Perou C.M. Lonning P.E. Brown P.O. Borresen-Dale A.L. Sorlie T., Tibshirani R. and Botstein D. Repeated observation of breast tumor subtypes in indepedent gene expression data sets. *Proc Natl Acad Sci USA*, 1àà(14):8418–8423, 2003.

[52] Tibshirani R. Aas T.-Geisher S. Johnsen H. Hastie T. Eisen M.B. van de Rijn M. Jeffrey S.S. Thorsen T. Quist H. Matese J.C. Brown P.O. Botstein D. Eystein Lonning P. Sorlie T., Perou C.M. and Borresen-Dale A.L. Gene expression patterns breast carcinomas distinguish tumor subclasses with clinical implications. *Proc. Matl. Acad. Sci. USA*, 98(19):10869–10874, 2001.

[53] McShane L.M. Korn E.L.-Long P.M. Jazaeri A. Martiat P. Fox S.B. Harris A.L. Sotiriou C., NEO S.Y. and Liu E.T. Breast cancer classification and prognosis based on gene expression profiles from a population-based study. *Proc. Natl. Acad. Sci.*, 100(18):10393–10398, 2003.

[54] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, 36(1):111–147, 1974.

[55] Hothorn T. Exact distributions for rank and permutation tests. R library, 2004.

[56] Spitznagel EL Jr Xu P Fu D Dimitrov DS Lempicki RA Raaka BM Tan PK, Downey TJ and Cam MC. Evaluation of gene expression measurements from commercial microarray platforms. *Nucleic Acids Res*, 31(19):5676–5684, 2003.

[57] J.W. Tukey. *Exploratory Data Analysis.* Addison, 1977.

[58] IBMM ULB and UMH. Biovallee. http://www.biovallee.be, 2003.

[59] W.N. Venables and B.D. Ripley. *Modern Applied Statistics with S.* Springer, 2002.

[60] Wang G Wang MD Fojo AT Sunshine M Narasimhan S Kane DW Reinhold WC Lababidi S Bussey KJ Riss J Barrett JC Zeeberg BR, Feng W and Weinstein JN. Gominer: A ressource for biological interpretation of genomic and proteomic data. *Genome Biology*, 4(4):R28, 2003.

Appendix A

Microarray Platforms

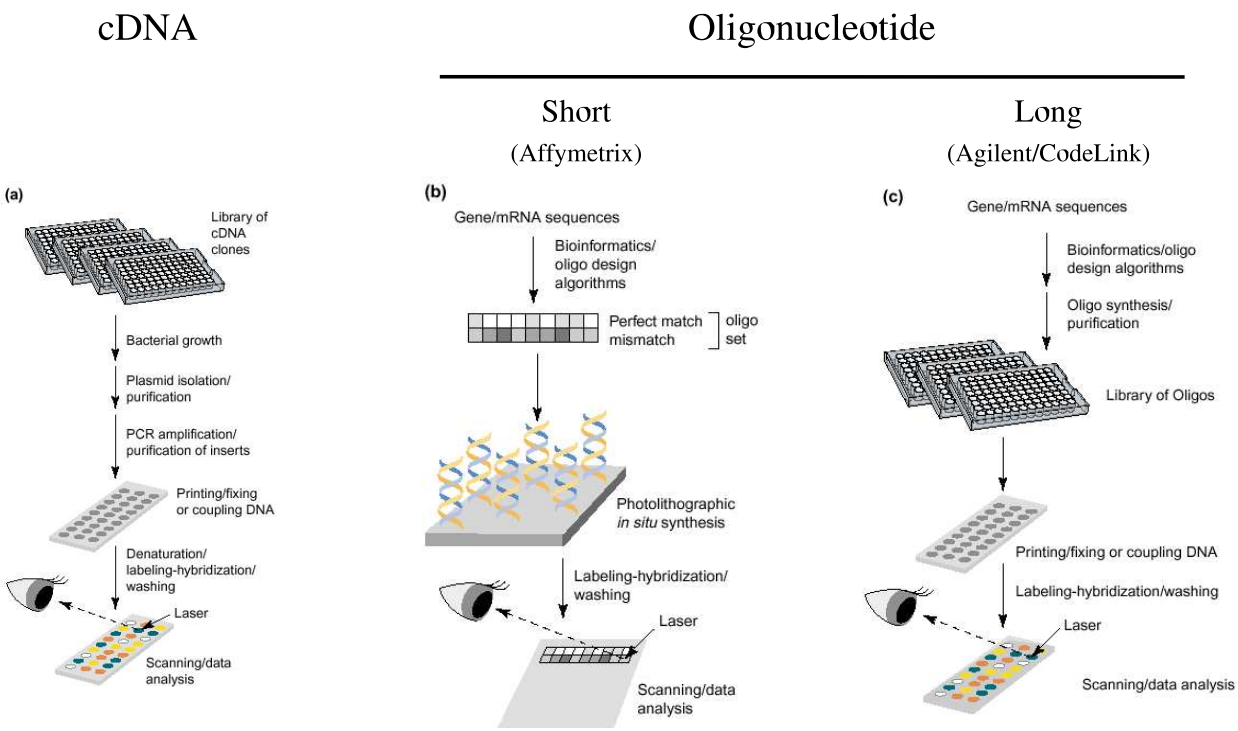


Figure A.1: Overview of different microarray platforms.

Appendix B

Analysis Functions

This appendix will introduce the basics of used functions for the analysis.

B.1 Read Data

The data are read through three different functions given in the file *medic_read.R*:

- medic.read: this function just reads all the *CEL* files in an AffyBatch object (see *affy* library)[1]. The AffyBatch object structure is described in the *affy* vignette (see Bioconductor help).
- medic.read.mas: this function reads and carries out the treatments to obtain expression measures (see the section B.2 for details). The function particularity is the possibility to read several subsets of *CEL* files and to concatenate them in a complete exprSet object[2]. The correctness of the this procedure is justified by the fact that all the steps are independent of the number of experiments (in the contrary of *rma* [7, 45, 44] which needs all the experiments to compute the normalization). It allows us to read all the data in a workstation with a reasonable amount of RAM.
- medic.read.justrma: this function reads and carries out the treatments (rma) to obtain expression measures (the details about these treatments can be found in the Bioconductor help).

Some string manipulation functions have been implemented to read correctly demographic data (prognosis and experiment names). The code of *medic_read.R* is given in appendix F.

Moreover, the chips hgu133a and hgu133b have to be read separately. There are 168 affy ids in common. Several methods have been implemented to handle this situation (see *medic.R*). Only the "mean" method have been used for the rest of the project, the mean between each affy id double is computed and is kept in the final exprSet object. The code is given in appendix G. This file contains the definitions of a set of useful functions for Affymetrix data analysis.

B.2 Get Expression Measure

All the treatments are carried out by the functions described above. Here are the details for each treatment.

[1] The memory space required to store all the probe values for the 99 patients is about 1,5 Gbytes.
[2] The exprSet object contains the expression measures for a set of microarray experiments.

B.2.1 MAS

The *medic.read.mas* uses the *mas* function for the background correction, the *constant* normalization, the *mas* function for the probe specific correction and the *mas* function for the summarization [8].

Background correction The background correction used in the function *medic.read.mas* is the *mas* implementation. This is an implementation of the background correction method outlined in the Statistical Algorithms Description Document [5]. The chip is broken into a grid of sixteen rectangular regions. For each region the lowest 2% of probe intensities are used to compute a background value for that grid. Each probe is then adjusted based upon a weighted average of the backgrounds for each of the regions. The weights are based on the distances between the location of the probe and the centroids of sixteen different regions. This method correct both PM and MM probes [8].

Normalization The *constant* normalization used in the function *medic.read.mas* is a scaling normalization. All the chips are scaled so that they have the same mean value. This would be typical of the approach taken by Affymetrix. However the Affymetrix normalization is usually done after summarization and this normalization is done before.

Probe Specific Correction The *mas* probe specific correction is used in the function *medic.read.mas*. An *ideal mismatch* is subtracted from PM. This mismatch value is the result of a small set of rules described in [5] and allows to avoid negative values of PM intensity which do not make physiological sense.

Summarization The *mas* summarization method is used in the function *medic.read.mas*. As documented in [5], a robust weighted average using one-step Tukey biweight [57] on $\log_2$ scale is performed on all corrected probe intensities in each probe set.

B.2.2 RMA

The *medic.read.justrma* uses the *rma* function for the background correction, the *quantile* normalization, the *pmonly* function for the probe specific correction and the *medianpolish* function for the summarization [8].

Background Correction The background correction used in the function *medic.read.justrma* is the *rma* implementation. The PM intensities are corrected by using a global model for the distribution of probe intensities. The model is suggested by looking at plots of the empirical distribution of probe intensities. In particular the observed PM probes are modeled as the sum of a normal noise component N (Normal with mean μ and variance σ^2) and an exponential signal component S (exponential with mean α). To avoid any possibility of negatives, the normal is truncated at zero. Given we have O the observed intensity, this then leads to an adjustment

$$E\left(s|O=o\right)=a+b\frac{\phi\left(\frac{a}{b}\right)-\phi\left(\frac{o-a}{b}\right)}{\Phi\left(\frac{a}{b}\right)-\Phi\left(\frac{o-a}{b}\right)-1}$$

where $a=s-\mu-\sigma^2\alpha$ and $b=\sigma$. Note that ϕ and Φ are the standard normal distribution density and distribution functions respectively.

The MM intensities are not corrected.

Normalization The normalization used in the function *medic.read.justrma* is the *quantile* implementation. The *quantile* was introduced by Bolstad *et al.* in 2003 [8, 9]. The goal is to give each chip the same empirical distribution. The algorithm 1 carries out this operation with X is the matrix pxn of probe intensities per array.

Algorithm 1 Quantile normalization [8].

1. Let X be a matrix pxn where p is the number of probes and n is the number of arrays
2. Sort each column of X to give X_{sort}
3. Take the means across the columns of X_{sort} and assign this mean to each element in the row to give X'_{sort}
4. Get $X_{normalized}$ by rearranging each column of X'_{sort} to have the same ordering as original X

The algorithm is illustrated in figure B.1.

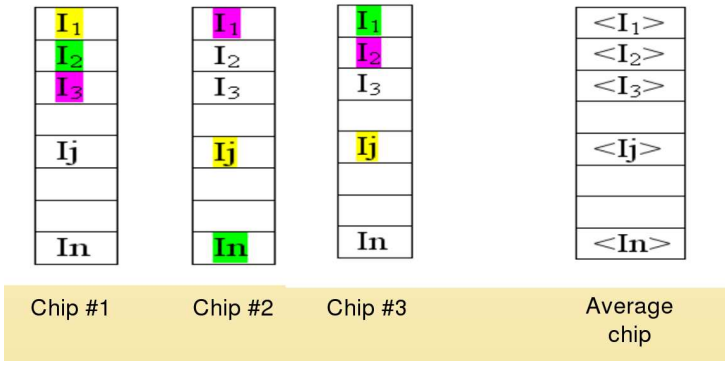


Figure B.1: Illustration of quantile normalization. Each color represents the same probe.

Probe Specific Correction The probe specific correction used in the function *medic.read.justrma* is the *pmonly* implementation. Actually only the PM intensities are kept.

Summarization The summarization used in the function *medic.read.justrma* is the *medianpolish* implementation. The *medianpolish* was introduced by Irizarry *et al.* in 2003 [19]. A multi-chip linear model is fit to data from each probe set. In particular for a probe set k with $1_k, 2_k, ..., I_k$ probes and data from $j = 1, 2, ..., J$, the following model is fitted

$$log_2\left(PM_{ij}^{(k)}\right) = \alpha_i^{(k)} + \beta_j^{(k)} + \epsilon_{ij}^{(k)}$$

where α_i is a probe effect and β_j is the log_2 expression value. The *medianpolish* is an algorithm [57] for fitting this model robustly. The calculated expression value is in log_2 scale.

B.3 Prefiltering

This step is skipped in this analysis design.

B.4 Filtering

This step is skipped in this analysis design.

B.5 Classification

Two classification techniques are used in this analysis design: the k-nearest neighbours (KNN) [46, 59] and the support vector machine (SVM) [12]. The pseudo-code of the classification is given in algorithm 3. The code of the *medic_simple_classif.R* is given in appendix H.

For the KNN and the SVM classifier, a structural identification and a feature selection steps are implemented:

- Structural identification [15] : the structural identification permits to tune the parameters (the structure) of the considered classifier to increase performance.
- Feature selection [54] : the feature selection permits to assess subsets of variables (i.e. the most discriminating genes for the classification) according to their usefulness to a given classifier.

Detailed descriptions about the classifiers, their structural identification, and the feature selection are given below.

B.5.1 KNN

The function *knn* in R (library *class*) can be used for classification. It uses the Euclidean distance to select the k-nearest neighbours and the classification is decided by majority vote, with ties broken at random. If there are ties for the k-th nearest vector, all candidates are included in the vote.

Structural identification The parameter k is chosen by a preliminary structural identification step. Actually a global loop with the structural identification could be done but will increase dramatically the computation time. All the patients are used to identify a sub-optimal k (selected between 3 and 30) and the first 100 ranked genes are used (be careful with these arbitrary criteria) in a L-O-O cross-validation procedure. This procedure can be lead to some overfitting because the choice of k is based on the whole dataset. However the global loop could be implemented with an efficient computer architecture (like a cluster).

B.5.2 SVM

The function *svm* in R (library *e1071*) can be used for classification. SVM's were developed by Cortes and Vapnick [13] for binary classification. Basically, the optimal separating hyperplane between the two classes is computed by maximizing the margin between the classes' closest points (see figure B.2). The points lying on the boundaries are called *support vectors*, and the middle of

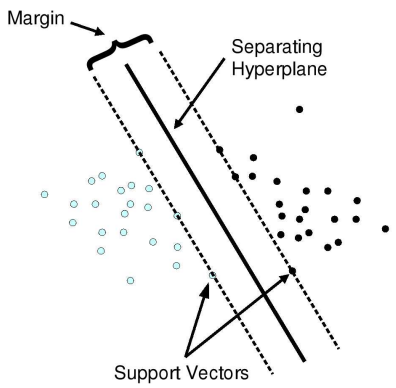


Figure B.2: Linear separable case of classification [34].

the margin is the optimal separating hyperplane.

The points on the "wrong" side of the discriminant margin are weighted down to reduce their influence. When a linear separator can not be found, the points are projected into a higher-dimensional space where the points effectively become linearly separable (see *kernel techniques*). A program able to perform such optimization tasks is called a *Support Vector Machine*.

The mode selected for the SVM is *C-classification* [34] with the radial basis function (RBF) [34] kernel because its good general performance and the few number of parameters (only two, the cost and the gamma parameters).

Structural identification Like the KNN, some parameters can be tuned, the cost and the gamma parameters. In [12], authors suggest to perform a grid search for the parameters:

- A large grid search is performed to detect interesting range of parameters:

$$cost = 2^{-15}...2^{13}2^{15} \rightarrow x_1$$

$$gamma = 2^{-15}...2^{3}2^{5} \rightarrow x_2$$

- A small grid search is performed to improve fine the classification performance:

$$cost = 2^{x_1-1}...2^{x_1+0.75}2^{X_1+1} \rightarrow cost_{tuned}$$

$$gamma = 2^{x_2-1}...2^{x_2+0.75}2^{x_2+1} \rightarrow gamma_{tuned}$$

As described previously, the structural identification is equivalent for the KNN and the SVM (L-O-O cross-validation procedure with the first 100 ranked genes). The grid search needs also a lot of computation time and a global structural identification loop would be too much for a common workstation.

B.5.3 Class Weights

We can see that the two populations (JRH and IGR) are not perfectly balanced: There are only 1/4 of patients who have developed distant metastases during the first five years. In clinical studies, one type of error is more important than another, the false negative rate. If we consider that that the relapse is positive and the absence of relapse is negative (see table B.1), the false negatives represent the patients for which the diagnosis suggests to not give a therapy but it is necessary in reality.

	Reality	
Prediction	relapse(+)	non-relapse(-)
relapse(+)	TP	FP
non-relapse(-)	FN	TN

Table B.1: Contingency table in our classification.

To avoid such a problem, the class weight parameter can be tuned. We consider than the class of relapses is more important (weight of 10) than the class of healthy patients (weight of 1).

B.5.4 Feature Selection

The feature selection described in algorithm 2 is a forward procedure. To control the exploration space of the selection, a specific method was introduced:when the added gene does not increase the classification performance more than some level (specified in parameter), the feature selection is stopped and the best set of genes is selected. Otherwise, even the classification performance decreased, the feature selection continues to search another sub-optimal set of marker genes.

The quality estimator q selected for the feature selection is given in equation B.1.

$$q = clw_{non-relapse} \#false\_positive + clw_{relapse} * \#false\_negative \quad \text{(B.1)}$$

where *clw* is the weight of the considered class. Even if the classifier is tuned to take into account the weights of each class, this quality estimator penalizes again the false negatives.

Algorithm 2 Pseudo-code of the feature selection (see *medic_simple_classif.R* in appendix H).

Compute the correlation between expression measures of each gene and the outcome on the training set
Rank the genes according to the absolute valor of their correlation coefficient
For each i from 1 to number of genes //feature selection
  Select the i first ranked genes as marker genes
  For each patient in the training set //leave-one-out for feature selection
    Evaluate the classification with the selected genes, for the patient left out
  endFor
  Stop the feature selection if no further improvement
  Save the best set of marker genes
endFor
Saved best set of marker genes contains the feature selected

B.5.5 Pseudo-code for the Whole Classification Procedure

Algorithm 3 Pseudo-code of the classification for the implemented analysis design (see *medic_simple_classif.R* in appendix H).

Structural identification for the selected classification technique
For each of n patients of the whole dataset //global leave-one-out
  Divide the whole dataset in training set (n-1 patients) and validation set (1 patient)
  Compute the correlation between expression measures of each gene and the outcome on the training set
  Rank the genes according to the absolute valor of their correlation coefficient
  For each i from 1 to number of genes //feature selection
    Select the i first ranked genes as marker genes
    For each patient in the training set //leave-one-out for feature selection
      Evaluate the classification for the patient left out
    endFor
    Stop the feature selection if no further improvement
    Save the best set of marker genes
  endFor
  Evaluate the classification using the whole training set and the feature selected, for the patient in the validation set
endFor
Evaluate the global classification rate

Appendix C

Classification With SVM

All the results shown in this section come from the SVM classifier. The following section and figures are identical to the KNN section 3.2. Only the remarks specific to SVM are noted.

C.1 Structural Identification

As described in paragraph B.5.2, the structural identification is performed with all the patients. The parameters, cost *and gamma*, re tested to find a sub-optimal values. We can see the evolution of the misclassification rate in figure 3.12.

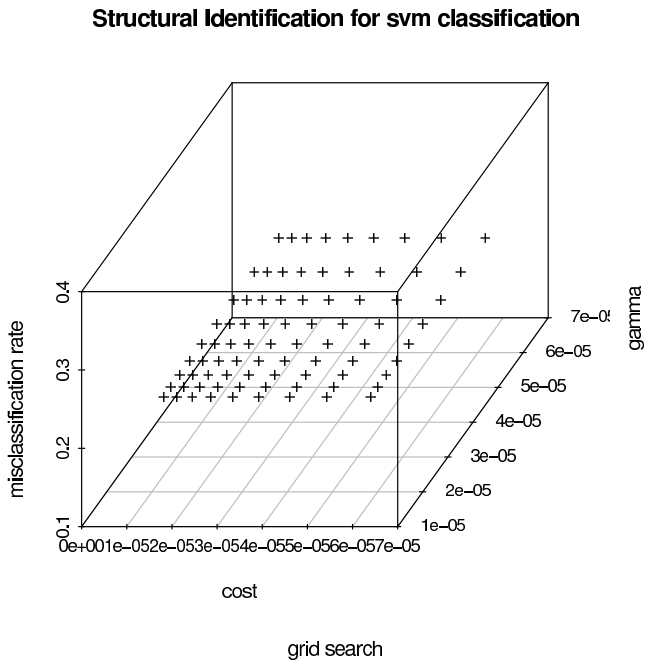


Figure C.1: Structural identification for the SVM classifier. The sub-optimal parameters, *cost* and *gamma*, equal to $1.5\,10^{-5}$ and $1.5\,10^{-5}$ respectively.

Like the KNN, only the global misclassification rate is used to assess the quality of the classifier. The code of *medic_ struct_ id.R* is given in appendix I.

C.2 Misclassification Rate during Feature Selection

We can see the evolution of the misclassification rate in the figure C.2. The profiles of figures for SVM are different to KNN but the misclassification rate is always very high. In comparison to the KNN classifier, the size of set of marker genes is smaller.

C.3 Marker Genes during Feature Selection

Because of the smaller size of set of marker genes for the SVM classifier, there is less selected genes during the feature selection and the frequency of each gene is smaller (maximum 3). The conclusion for the KNN classifier is the same in the case of the SVM classifier.

C.4 Misclassification Rate

The misclassification rate is approximatively the same for the false positives and the false negatives, very poor. This classifier is not really valuable: The trend is to classify patients in the relapse class in most of the cases.

The misclassification rate computed by the global leave-one-out is 65/75 false positives and 2/24 false negatives. The SVM classifies better the relapse class than the KNN on the contrary of the non-relapse class. This is due to the quality parameter q which is too stringent for the SVM classifier. The performance could be improved by selecting a more adapted values for the class weights.

C.5 Signature of Marker Genes

After the performance evaluation of the classification technique by leave-one-out, a signature of marker genes can be computed from all the patients. The feature selection is carried out to determine the marker genes (see figure C.5).

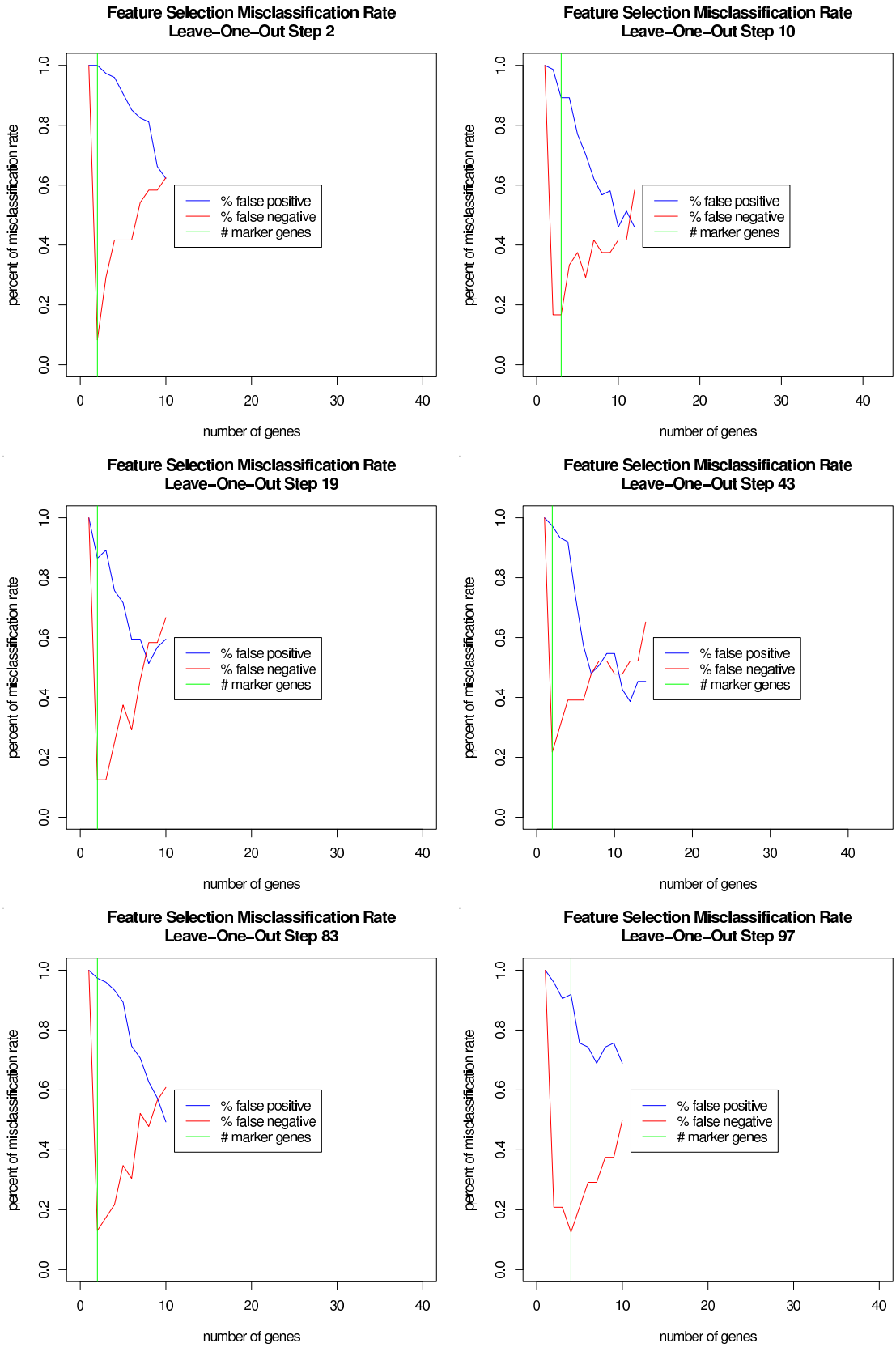


Figure C.2: Subset of figures which represent the evolution of misclassification rate during the feature selection in SVM classifier. There are 99 figures but only few of them are shown here.

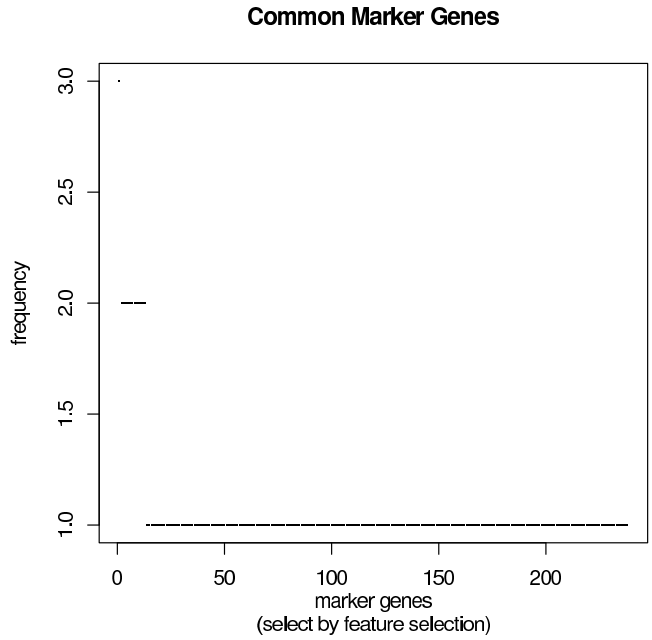
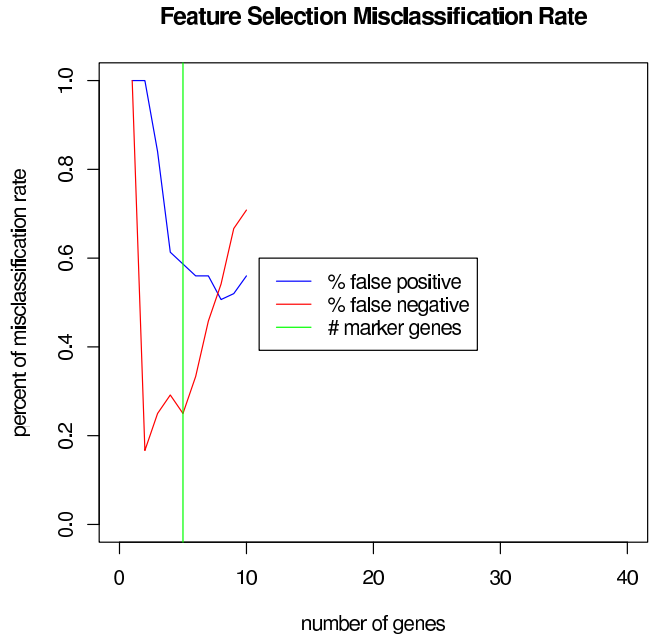
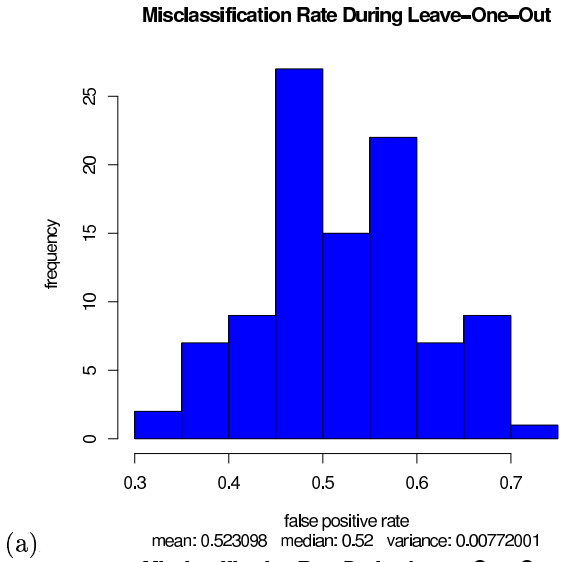


Figure C.3: Marker genes selected during the feature selection for the SVM classifier.

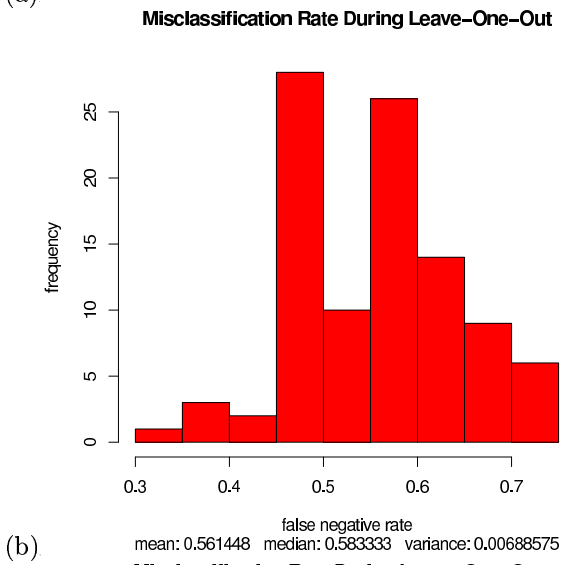


5 marker genes: **224529_s_at**, **223176_at**, **227921_at**, **238353_at**, and **244129_at**

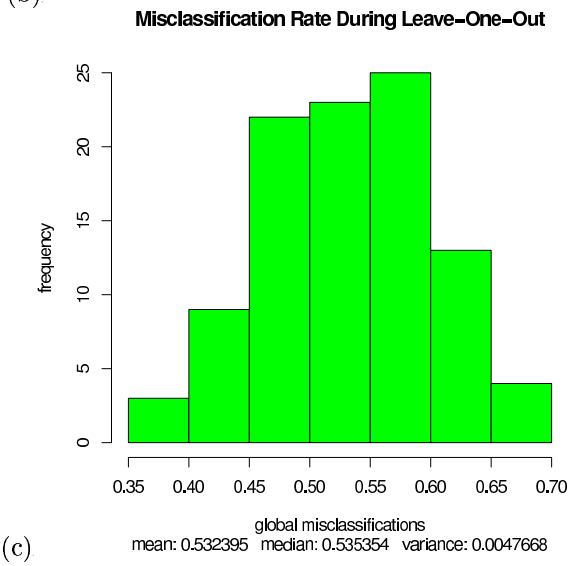
Figure C.5: Feature selection for the SVM classifier to determine the signature of marker genes.



(a)



(b)



(c)

Figure C.4: (a) Misclassification rate for the false positives. (b) Misclassification rate for the false negatives. (c) Global misclassification rate.

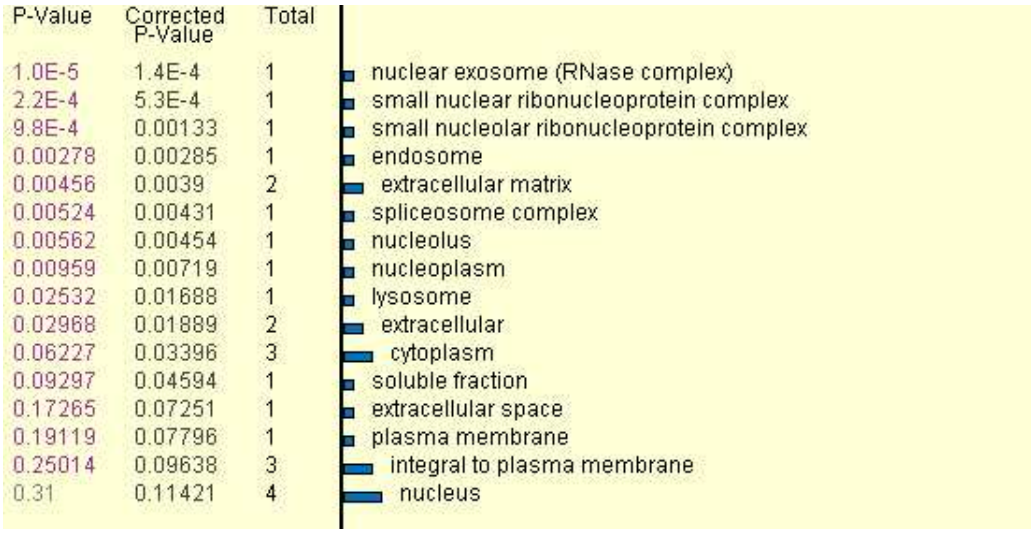
Appendix D

Gene Ontology in Marker Gene Signature

An example of the usefulness of GO tools like Onto-Express is given here. The list of marker genes comes from a signature with the KNN classifier in considering only the chip hgu133a. The signature contains 42 marker genes. The results for the three GO categories (biological process, cellular component, and molecular function) are represented graphically (see figures D.1, D.2, and D.3).

P-Value	Corrected P-Value	Total	
0.0	4.0E-5	1	copulation
1.0E-5	1.2E-4	1	tyrosine catabolism
3.0E-5	1.2E-4	1	DNA catabolism
6.0E-5	2.6E-4	1	aromatic amino acid family metabolism
9.0E-5	3.2E-4	1	phenylalanine catabolism
2.2E-4	5.0E-4	1	peripheral nervous system development
3.1E-4	6.1E-4	1	inactivation of MAPK
3.7E-4	6.6E-4	2	defense response
4.2E-4	7.1E-4	1	base-excision repair
5.4E-4	8.7E-4	1	spliceosome assembly
6.0E-4	8.8E-4	1	G-protein signaling, coupled to IP3 second messenger (phospholipase C activating)
0.00114	0.0015	2	antimicrobial humoral response (sensu Vertebrata)
0.0020	0.00248	1	response to virus
0.00225	0.0027	1	xenobiotic metabolism
0.00264	0.00292	1	rRNA processing
0.00278	0.00278	1	excretion
0.00307	0.00293	1	olfaction
0.00336	0.00307	1	cell shape and cell size control
0.00839	0.00653	1	cell cycle arrest
0.01305	0.00945	1	homophilic cell adhesion
0.01333	0.00949	1	nonselective vesicle transport
0.01633	0.01124	1	DNA replication
0.02723	0.01787	1	protein amino acid dephosphorylation
0.03289	0.02031	1	cell motility
0.03807	0.02284	1	carbohydrate metabolism
0.04032	0.02352	1	protein folding
0.05277	0.02995	1	inflammatory response
0.07025	0.03643	1	synaptic transmission
0.08517	0.04258	1	regulation of cell cycle
0.09297	0.0454	1	regulation of transcription from Pol II promoter
0.10724	0.05004	1	neurogenesis
0.13804	0.06234	1	cell proliferation
0.14143	0.06252	1	G-protein coupled receptor protein signaling pathway
0.19551	0.07895	1	immune response
0.24076	0.09363	1	development
0.29098	0.10912	1	protein amino acid phosphorylation
0.32323	0.11804	1	signal transduction
0.37867	0.13478	2	regulation of transcription, DNA-dependent

Figure D.1: Biological process category for the signature composed by the 42 marker genes by using Onto-Express [30].



P-Value	Corrected P-Value	Total	
1.0E-5	1.4E-4	1	nuclear exosome (RNase complex)
2.2E-4	5.3E-4	1	small nuclear ribonucleoprotein complex
9.8E-4	0.00133	1	small nucleolar ribonucleoprotein complex
0.00278	0.00285	1	endosome
0.00456	0.0039	2	extracellular matrix
0.00524	0.00431	1	spliceosome complex
0.00562	0.00454	1	nucleolus
0.00959	0.00719	1	nucleoplasm
0.02532	0.01688	1	lysosome
0.02968	0.01889	2	extracellular
0.06227	0.03396	3	cytoplasm
0.09297	0.04594	1	soluble fraction
0.17265	0.07251	1	extracellular space
0.19119	0.07796	1	plasma membrane
0.25014	0.09638	3	integral to plasma membrane
0.31	0.11421	4	nucleus

Figure D.2: Cellular component category for the signature composed by the 42 marker genes by using Onto-Express [30].

P-Value	Corrected P-Value	Total	
0.0	8.0E-5	1	4-hydroxyphenylpyruvate dioxygenase activity
0.0	1.0E-4	1	epsilon DNA polymerase activity
0.0	8.0E-5	1	pyrimidine-specific mismatch base pair DNA N-glycosylase activity
1.0E-5	9.0E-5	1	calcium sensitive guanylate cyclase activator activity
1.0E-5	1.0E-4	1	fatty-acyl-CoA synthase activity
3.0E-5	1.4E-4	1	deoxyribonuclease activity
1.2E-4	3.9E-4	1	histamine receptor activity
1.2E-4	3.6E-4	1	MAP kinase phosphatase activity
1.5E-4	4.0E-4	1	antifungal peptide activity
1.5E-4	4.3E-4	1	interferon-alpha/beta receptor binding
1.8E-4	4.6E-4	1	prenylated protein tyrosine phosphatase activity
2.2E-4	4.8E-4	1	extracellular matrix glycoprotein
3.1E-4	6.3E-4	1	insulin receptor binding
3.7E-4	6.7E-4	1	3'-5' exoribonuclease activity
5.4E-4	8.4E-4	1	protein serine/threonine phosphatase activity
9.0E-4	0.00126	1	exonuclease activity
0.00155	0.00198	1	olfactory receptor activity
0.00251	0.00293	1	antiviral response protein activity
0.00264	0.0030	1	endonuclease activity
0.00278	0.00292	1	hydrolase activity, acting on glycosyl bonds
0.00307	0.00299	1	damaged DNA binding
0.00321	0.0030	1	transcription cofactor activity
0.00417	0.00372	1	protein kinase activity
0.00434	0.00379	1	enzyme activator activity
0.00487	0.00409	1	protein tyrosine phosphatase activity
0.00793	0.00628	1	serine protease inhibitor activity
0.00851	0.0065	2	transcription co-activator activity
0.0154	0.01078	1	pre-mRNA splicing factor activity
0.04735	0.02724	1	RNA polymerase II transcription factor activity
0.05892	0.03299	1	rhodopsin-like receptor activity
0.0664	0.03575	1	structural molecule activity
0.07004	0.03677	2	RNA binding
0.07448	0.03814	3	hydrolase activity
0.09522	0.04597	3	transcription factor activity
0.11298	0.05214	1	cell adhesion molecule activity
0.14007	0.06258	1	protein serine/threonine kinase activity
0.15308	0.06697	1	oxidoreductase activity
0.17124	0.07264	1	zinc ion binding
0.19094	0.07862	3	DNA binding
0.32777	0.11867	1	protein binding
0.38237	0.13495	1	ATP binding
0.48713	0.1677	1	transferase activity

Figure D.3: Molecular function category for the signature composed by the 42 marker genes by using Onto-Express [30].

Appendix E

Code of medic_qc.R

```
#quality control for the Medic project data
library(affy);
library(simpleaffy);
library(AffyExtensions);
setwd("/home/bahamut/project_medic/R_scripts/");
source("medic_sources.R");
med <- medic();
#oxford
abatch.chipA <- med$medic.read("oxford", "../raw_data/oxford/medic_oxford_HG-U133A/",
"../raw_data/oxford/Demographics_oxford_distant+local.csv");
abatch.chipB <- med$medic.read("oxford", "../raw_data/oxford/medic_oxford_HG-U133B/",
"../raw_data/oxford/Demographics_oxford_distant+local.csv");
#IGR
abatch.chipA <- med$medic.read("igr", "../raw_data/igr/medic_igr_HG-U133A/",
"../raw_data/igr/Demographics_igr_distant+loc.csv");
abatch.chipB <- med$medic.read("igr", "../raw_data/igr/medic_igr_HG-U133B/",
"../raw_data/igr/Demographics_igr_distant+loc.csv");
#karolinska68
abaabatchA.k68 <- med$medic.read("karolinska68", "../raw_data/karolinska/karolinska_68/A/",
"../raw_data/karolinska/karolinska_68/Demographics_karolinska_68_relapse.csv");
abatchB.k68 <- med$medic.read("karolinska68", "../raw_data/karolinska/karolinska_68/B/",
"../raw_data/karolinska/karolinska_68/Demographics_karolinska_68_relapse.csv")
nbr.exp <- length(sampleNames(abatch.chipB));
nbr.gene <- length(geneNames(abatch.chipB));
##############################################################################
#histogram of PM intensities
pdf("pm_hist_chipA.pdf");
#chip A
main <- sprintf("IGR Population - Relapse\n(%i patients on chip %s)",
as.integer(length(sampleNames(abatch.chipA))), annotation(abatch.chipA));
sub <- sprintf("histogram of PM intensities");
plotDensity.AffyBatch(abatch.chipA, which="pm");
title(main=main, sub=sub);
sub <- sprintf("histogram of MM intensities");
plotDensity.AffyBatch(abatch.chipA, which="mm");
title(main=main, sub=sub);
sub <- sprintf("histogram of PM-MM intensities");
plotDensity.AffyBatch(abatch.chipA, which="both");
title(main=main, sub=sub);
dev.off();
#chip B
pdf("pm_hist_chipB.pdf");
main <- sprintf("IGR Population - Relapse\n(%i patients on chip %s)",
as.integer(length(sampleNames(abatch.chipB))), annotation(abatch.chipB));
sub <- sprintf("histogram of PM intensities");
plotDensity.AffyBatch(abatch.chipB, which="pm");
title(main=main, sub=sub);
```

```
sub <- sprintf("histogram of MM intensities");
plotDensity.AffyBatch(abatch.chipB, which="mm");
title(main=main, sub=sub);
sub <- sprintf("histogram of PM-MM intensities");
plotDensity.AffyBatch(abatch.chipB, which="both");
title(main=main, sub=sub);
dev.off();
################################################################################
#image of arrays
png("image_chipA_igr%03d.png");
par(mfrow=c(1,1));
image(abatch.chipA);
dev.off();
png("image_chipB_igr%03d.png");
par(mfrow=c(1,1));
image(abatch.chipB);
dev.off();
################################################################################
#boxplot of arrays
pdf("boxplot_chipA_igr.pdf");
main <- sprintf("Igr Population - Relapse\n(%i patients on chip %s)",
as.integer(length(sampleNames(abatch.chipA))), annotation(abatch.chipA));
sub <- sprintf("boxplot of arrays");
boxplot(abatch.chipA, col=2:4);
title(main=main, sub=sub);
dev.off();
pdf("boxplot_chipB_igr.pdf");
main <- sprintf("Igr Population - Relapse\n(%i patients on chip %s)",
as.integer(length(sampleNames(abatch.chipB))), annotation(abatch.chipB));
sub <- sprintf("boxplot of arrays");
boxplot(abatch.chipB, col=2:4);
title(main=main, sub=sub);
dev.off();
################################################################################
#RNA degradation
degA <- AffyRNAdeg(abatch.chipA);
#summaryAffyRNAdeg(degA);
#compute the ratio between the intensity in the probe at the 5' side and the 3' side
ratio.rnadeg.A <- NULL;
for (i in 1:nrow(degA$means.by.number)) {
        ratio.rnadeg.A <- c(ratio.rnadeg.A, degA$means.by.number[i,1] /
degA$means.by.number[i,ncol(degA$means.by.number)]);
}
degB <- AffyRNAdeg(abatch.chipB);
#summaryAffyRNAdeg(degB);
#compute the ratio between the intensity in the probe at the 5' side and the 3' side
ratio.rnadeg.B <- NULL;
for (i in 1:nrow(degB$means.by.number)) {
        ratio.rnadeg.B <- c(ratio.rnadeg.B, degB$means.by.number[i,1] /
degB$means.by.number[i,ncol(degB$means.by.number)]);
}
pdf("rnadeg_chipA_igr.pdf");
main <- sprintf("\n\n\nIgr Population - Relapse\n(%i patients on chip %s)",
as.integer(length(sampleNames(abatch.chipA))), annotation(abatch.chipA));
plotAffyRNAdeg(degA, cols=1:nbr.exp);
title(main=main);
ylab <- "ratio betwen intensities 5'/3'";
xlab <- "samples";
plot(ratio.rnadeg.A, main="", xlab=xlab, ylab=ylab, col="blue")
stat.info <- sprintf("\n\nmean: %.5g   median: %.5g   std: %.5g   range: [%.5g,%.5g]",
mean(ratio.rnadeg.A, na.rm=TRUE), median(ratio.rnadeg.A, na.rm=TRUE),
sd(ratio.rnadeg.A, na.rm=TRUE), min(ratio.rnadeg.A), max(ratio.rnadeg.A));
title(main=main, sub=stat.info);
dev.off();
pdf("rnadeg_chipB_igr.pdf");
```

```
main <- sprintf("\n\n\nIgr Population - Relapse\n(%i patients on chip %s)",
as.integer(length(sampleNames(abatch.chipB))), annotation(abatch.chipB));
plotAffyRNAdeg(degB, cols=1:nbr.exp);
title(main=main);
plot(ratio.rnadeg.B, main="", xlab=xlab, ylab=ylab, col="blue")
stat.info <- sprintf("\n\nmean: %.5g   median: %.5g   std: %.5g   range: [%.5g,%.5g]",
mean(ratio.rnadeg.B, na.rm=TRUE), median(ratio.rnadeg.B, na.rm=TRUE),
sd(ratio.rnadeg.B, na.rm=TRUE), min(ratio.rnadeg.B), max(ratio.rnadeg.B));
title(main=main, sub=stat.info);
dev.off();
################################################################################
#RawQ values using the RPT files
pdf("rawq.pdf");
pop <- "igr";
xlab <- "sample";
ylab <- "RawQ";
my.dir <- "/mnt/bordet/cold/IGR/Medic(IGR)_HG-U133B_COLONNE/";
main <- sprintf("Igr Population - Relapse\n(%i patients on chip %s)",
as.integer(length(sampleNames(abatch.chipB))), annotation(abatch.chipB));
rpt.files <- dir(dir(my.dir[j], full.names=TRUE), full.names=TRUE, pattern="*.RPT");
rpt.names <- dir(dir(my.dir[j], full.names=TRUE), pattern="*.RPT");
rawq <- NULL;
for (i in 1:length(rpt.files)) {
        rawq <- c(rawq, readLines(rpt.files[i])[grep("RawQ", readLines(rpt.files[i]))]);
}
rawq <- as.numeric(substr(rawq, 15, nchar(rawq)));
names(rawq) <- paste(rpt.names, pop, sep="_");
plot(rawq, main="", xlab=xlab, ylab=ylab, col="blue")
stat.info <- sprintf("\n\nmean: %.5g   median: %.5g   std: %.5g   range: [%.5g,%.5g]",
mean(rawq, na.rm=TRUE), median(rawq, na.rm=TRUE), sd(rawq, na.rm=TRUE), min(rawq), max(rawq));
title(main=main[j], sub=stat.info);
dev.off();
################################################################################
#average background using the RPT files
my.dir <- "/mnt/bordet/cold/IGR/Medic(IGR)_HG-U133B_COLONNE/";
main <- sprintf("Igr Population - Relapse\n(%i patients on chip %s)",
as.integer(length(sampleNames(abatch.chipB))), annotation(abatch.chipB));
rpt.files <- dir(dir(my.dir[j], full.names=TRUE), full.names=TRUE, pattern="*.RPT");
rpt.names <- dir(dir(my.dir[j], full.names=TRUE), pattern="*.RPT");
avg.bg <- NULL;
for (i in 1:length(rpt.files)) {
        info.temp <- readLines(rpt.files[i])[grep("Background", readLines(rpt.files[i]))+1];
        info.temp <- unlist(strsplit(info.temp, "\t"));
        info.temp <- info.temp[-1];
        info.temp <- unlist(strsplit(info.temp, "[ ]"));
        info.temp <- as.numeric(info.temp[c(2,4,6,8)]);
        names(info.temp) <- c("avg", "std", "min", "max");
        #only the average background
        avg.bg <- c(avg.bg, info.temp[1]);
}
names(avg.bg) <- paste(rpt.names, pop, sep="_");
pdf("avg_background.pdf");
pop <- "igr";
xlab <- "sample";
ylab <- "average background";
plot(avg.bg, main="", xlab=xlab, ylab=ylab, col="blue")
stat.info <- sprintf("\n\nmean: %.5g   median: %.5g   std: %.5g   range: [%.5g,%.5g]",
mean(avg.bg, na.rm=TRUE), median(avg.bg, na.rm=TRUE), sd(avg.bg, na.rm=TRUE), min(avg.bg), max(avg.bg));
title(main=main[j], sub=stat.info);
dev.off();
################################################################################
#scaling factor using the RPT files
my.dir <- "/mnt/bordet/cold/IGR/Medic(IGR)_HG-U133B_COLONNE/";
main <- sprintf("Igr Population - Relapse\n(%i patients on chip %s)",
as.integer(length(sampleNames(abatch.chipB))), annotation(abatch.chipB));
rpt.files <- dir(dir(my.dir[j], full.names=TRUE), full.names=TRUE, pattern="*.RPT");
```

```
rpt.names <- dir(dir(my.dir[j], full.names=TRUE), pattern="*.RPT");
scf <- NULL;
for (i in 1:length(rpt.files)) {
        scf <- c(scf, readLines(rpt.files[i])[grep("SF", readLines(rpt.files[i]))]);
}
scf <- as.numeric(substr(scf, 19, nchar(scf)));
names(scf) <- paste(rpt.names, pop, sep="_");
pdf("scaling_factor.pdf");
pop <- "igr";
xlab <- "sample";
ylab <- "scaling factor";
plot(scf, main="", xlab=xlab, ylab=ylab, col="blue")
stat.info <- sprintf("\n\nmean: %.5g   median: %.5g   std: %.5g   range: [%.5g,%.5g]",
mean(scf, na.rm=TRUE), median(scf, na.rm=TRUE), sd(scf, na.rm=TRUE), min(scf), max(scf));
title(main=main[j], sub=stat.info);
dev.off();
###############################################################################
#compute the Present/Absent/Marginal calls
#calls in the 'exprs' and p-value in the 'se.exprs'
abatch.mas5calls <- mas5calls(abatch);
abatch.chipA.mas5calls <- abatch.mas5calls;
save(abatch.chipA.mas5calls, file="abatch_chipA_all_mas5calls.RData");
#percent of P/A/M
p.prct <- apply(exprs(abatch.mas5calls)=="P", 2, sum)/nbr.gene;
a.prct <- apply(exprs(abatch.mas5calls)=="A", 2, sum)/nbr.gene;
m.prct <- apply(exprs(abatch.mas5calls)=="M", 2, sum)/nbr.gene;
pdf("mas5calls.pdf");
xlab <- "samples";
main <- sprintf("Igr Population - Relapse\n(%i patients on chip %s)",
as.integer(length(sampleNames(abatch))), annotation(abatch));
ylab <- "percentage of Present calls";
stat.info <- sprintf("\n\nmean: %.5g   median: %.5g   std: %.5g   range: [%.5g,%.5g]", mean(p.prct, na.rm=TRUE),
median(p.prct, na.rm=TRUE), sd(p.prct, na.rm=TRUE), min(p.prct), max(p.prct));
plot(p.prct, col="blue", main="", xlab=xlab, ylab=ylab);
title(main=main, sub=stat.info);
ylab <- "percentage of Absent calls";
stat.info <- sprintf("\n\nmean: %.5g   median: %.5g   std: %.5g   range: [%.5g,%.5g]", mean(a.prct, na.rm=TRUE),
median(a.prct, na.rm=TRUE), sd(a.prct, na.rm=TRUE), min(a.prct), max(a.prct));
plot(a.prct, col="blue", main="", xlab=xlab, ylab=ylab);
title(main=main, sub=stat.info);
ylab <- "percentage of Marginal calls";
stat.info <- sprintf("\n\nmean: %.5g   median: %.5g   std: %.5g   range: [%.5g,%.5g]", mean(m.prct, na.rm=TRUE),
median(m.prct, na.rm=TRUE), sd(m.prct, na.rm=TRUE), min(m.prct), max(m.prct));
plot(m.prct, col="blue", main="", xlab=xlab, ylab=ylab);
title(main=main, sub=stat.info);
dev.off();
###############################################################################
#control genes
#housekeeping controls
hs.control <- c("HUMISGF3A/M97935", "HUMRGE/M10098", "HUMGAPDH/M33197", "HSAC07/X00351", "M27830");
#search the exact probe set id
hs.id <- NULL;
for (i in 1:length(hs.control)) {
        #print(grep(hs.control[i], geneNames(abatch.chipB)));
        hs.id <- c(hs.id, list(name.control=hs.control[i], index.control=grep(hs.control[i], geneNames(abatch.chi
}
#spike controls
sp.control <- c("BioB", "BioC", "BioDn", "CreX", "DapX", "LysX", "PheX", "ThrX", "TrpnX");
#search the exact probe set id
sp.id <- NULL;
for (i in 1:length(sp.control)) {
        #print(grep(sp.control[i], geneNames(abatch.chipB)));
        sp.id <- c(sp.id, list(name.control=sp.control[i], index.control=grep(sp.control[i], geneNames(abatch.chi
}
#summary for control genes
#
```

```
#chip hgu133a and hgu133b
#
#HUMISGF3A/M97935:      AFFX-HUMISGF3A/M97935_3_at (22234, 22596)
#                               AFFX-HUMISGF3A/M97935_5_at (2235, 22597)
#
#HUMRGE/M10098: AFFX-HUMRGE/M10098_3_at (22238, 22600)
#                               AFFX-HUMRGE/M10098_5_at (22239, 22601)
#
#HUMGAPDH/M33197:       AFFX-HUMGAPDH/M33197_3_at (22231, 22593)
#                               AFFX-HUMGAPDH/M33197_5_at (22232, 22594)
#
#HSAC07/X00351: AFFX-HSAC07/X00351_3_at (22228, 22590)
#                               AFFX-HSAC07/X00351_5_at (22229, 22591)
#
#M27830:                        AFFX-M27830_3_at (22244, 22606)
#                               AFFX-M27830_5_at (22245, 22607)
#summary for spike control
#
#chip hgu133a and hgu133b
#
#BioB:                  AFFX-BioB-3_at (22216, 22578)
#                               AFFX-BioB-5_at (22217, 22579)
#
#BioC:                  AFFX-BioC-3_at (22219, 22581)
#                               AFFX-BioC-5_at (22220, 22582)
#
#BioDn:                 AFFX-BioDn-3_at (22221, 22583)
#                               AFFX-BioDn-5_at (22222, 22584)
#
#CreX:                  AFFX-CreX-3_at (22223, 22585)
#                               AFFX-CreX-5_at (22224, 22586)
#
#DapX:                  AFFX-DapX-3_at (22225, 22587)
#                               AFFX-DapX-5_at (22226, 22588)
#
#LysX:                  AFFX-LysX-3_at (22241, 22603)
#                               AFFX-LysX-5_at (22242, 22604)
#
#PheX:                  AFFX-PheX-3_at (22247, 22609)
#                               AFFX-PheX-5_at (22248, 22610)
#
#ThrX:                  AFFX-ThrX-3_at (22250, 22612)
#                               AFFX-ThrX-5_at (22251, 22613)
#
#TrpnX:                 AFFX-TrpnX-3_at (22253, 22615)
#                               AFFX-TrpnX-3_at (22254, 22616)
#exact names for control genes
hs <- c("AFFX-HUMISGF3A/M97935_3_at", "AFFX-HUMISGF3A/M97935_5_at",
"AFFX-HUMRGE/M10098_3_at", "AFFX-HUMRGE/M10098_5_at", "
AFFX-HUMGAPDH/M33197_3_at", "AFFX-HUMGAPDH/M33197_5_at",
"AFFX-HSAC07/X00351_3_at", "AFFX-HSAC07/X00351_5_at", "AFFX-M27830_3_at", "AFFX-M27830_5_at");
#exact names for spike controls
sp <- c("AFFX-BioB-3_at", "AFFX-BioB-5_at", "AFFX-BioC-3_at", "AFFX-BioC-5_at",
"AFFX-BioDn-3_at", "AFFX-BioDn-5_at", "AFFX-CreX-3_at", "AFFX-CreX-5_at",
"AFFX-DapX-3_at", "AFFX-DapX-5_at", "AFFX-LysX-3_at", "AFFX-LysX-5_at",
"AFFX-PheX-3_at", "AFFX-PheX-5_at", "AFFX-ThrX-3_at", "AFFX-ThrX-5_at", "AFFX-TrpnX-3_at", "AFFX-TrpnX-5_at");
#load mas5 expression set
load("../medic_quality_control/igr/chipA/esetA_mas_igr.RData");
#load mas5 calls
load("../medic_quality_control/igr/chipA/abatch_chipA_all_mas5calls.RData");
pdf("affy_control.pdf");
#for each experiment
for (j in 1:length(sampleNames(abatch.chipA.mas5calls))) {
        main <- sprintf("IGR Population - Relapse (chip %s)\n%s",
annotation(esetA.mas.igr), as.character(pData(esetA.mas.igr)[j,2]));
        #for each housekeeping control
```

```
hs.calls <- NULL;
hs.sig <- NULL;
for(i in 1:length(hs)) {
        hs.calls <- c(hs.calls, exprs(abatch.chipA.mas5calls)[geneNames(abatch.chipA.mas5calls) == hs[i],
        hs.sig <- c(hs.sig, exprs(esetA.mas.igr)[geneNames(esetA.mas.igr) == hs[i],j]);
}
#generate the indices
ind <- NULL;
for(i in 1:(length(hs)/2)) {
        ind <- c(ind,rep(i, 2));
}
#compute the colors
hs.colors <- NULL;
for(i in 1:length(hs)) {
        if (hs.calls[i] == "P") {
                if (i%%2 == 0) { #3' present
                        hs.colors <- c(hs.colors, "blue");
                }
                else { #5' present
                        hs.colors <- c(hs.colors, "red");
                }
        }
        else {
                if (i%%2 == 0) { #3' absent
                        hs.colors <- c(hs.colors, "purple");
                }
                else { #5' absent
                        hs.colors <- c(hs.colors, "orange");
                }
        }
}
ylab <- "signals";
xlab <-"housekeeping controls";
plot(ind, hs.sig, col=hs.colors, main="", xlab=xlab, ylab=ylab, xlim=c(0, (length(hs)/2)+7), ylim=c(0, 80
title(main=main);
legend(x=(length(hs)/2)+1, y=4000, legend=c("signal 3' (P)",
"signal 3' (A)", "signal 5' (P)", "signal 5' (A)"), col=c("red", "orange", "blue", "purple"), pch='o');
ylab <- "signal (3'/5')"
xlab <- "housekeeping controls";
plot(hs.sig[(1:length(hs.sig))%%2 == 1] / hs.sig[(1:length(hs.sig))%%2 == 0], xlab=xlab, ylab=ylab, main=
title(main=main);
#for each spike control
sp.calls <- NULL;
sp.sig <- NULL;
for(i in 1:length(sp)) {
        sp.calls <- c(sp.calls, exprs(abatch.chipA.mas5calls)[geneNames(abatch.chipA.mas5calls) == sp[i],
        sp.sig <- c(sp.sig, exprs(esetA.mas.igr)[geneNames(esetA.mas.igr) == sp[i],j]);
}

#generate the indices
ind <- NULL;
for(i in 1:(length(sp)/2)) {
        ind <- c(ind,rep(i, 2));
}
#compute the colors
sp.colors <- NULL;
for(i in 1:length(sp)) {
        if (sp.calls[i] == "P") {
                if (i%%2 == 0) { #3' present
                        sp.colors <- c(sp.colors, "blue");
                }
                else { #5' present
                        sp.colors <- c(sp.colors, "red");
                }
        }
        else {
                if (i%%2 == 0) { #3' absent
```

```
                                sp.colors <- c(sp.colors, "purple");
                        }
                        else { #5' absent
                                sp.colors <- c(sp.colors, "orange");
                        }
                }
        }
        ylab <- "signals";
        xlab <-"spike controls";
        plot(ind, sp.sig, col=sp.colors, main="", xlab=xlab, ylab=ylab, xlim=c(0, (length(sp)/2)+7), ylim=c(0, 85
        title(main=main);
        legend(x=(length(sp)/2)+1, y=4000, legend=c("signal 3' (P)",
"signal 3' (A)", "signal 5' (P)", "signal 5' (A)"), col=c("red", "orange", "blue", "purple"), pch='o');
        ylab <- "signal (3'/5')"
        xlab <- "spike controls";
        plot(sp.sig[(1:length(sp.sig))%%2 == 1] / sp.sig[(1:length(sp.sig))%%2 == 0], xlab=xlab, ylab=ylab);
        title(main=main);
}
dev.off();
################################################################################
#use of simpleaffy package
#scaling factor, average background, housekeeping controls and spike controls
abatch <- abatch.chipB;
qc <- qc.stats(abatch);
sf <- qc[,colnames(qc)=="s.f."];
gapdh35 <- qc[,"GAPDH.3'/5'"];
gapdh3M <- qc[,"GAPDH.3'/M"];
beta.actin35 <-qc[,"beta.actin.3'/5'"];
beta.actin3M <- qc[,"beta.actin.3'/M"];
avg.bg <- qc[,"avbg"];
spike <- qc[, c("bioB", "bioC", "bioD", "creX")];
spike[spike == 1] <- "P";
spike[spike != "P"] <- "A/M";
main <- sprintf("IGR Population - Relapse\n(%i patients on chip %s)",
as.integer(length(sampleNames(abatch))), annotation(abatch));
pdf("scaling_factor.pdf");
pop <- "igr";
xlab <- "sample";
ylab <- "scaling factor";
col <- rep("blue", nbr.exp);
col[sf > median(sf)*sqrt(3)] <- "red";
col[sf < median(sf)/sqrt(3)] <- "red";
plot(sf, ylim=c(0,2), main="", xlab=xlab, ylab=ylab, col=col)
abline(h=median(sf), col="black");
abline(h=median(sf)*sqrt(3), col="green");
abline(h=median(sf)/sqrt(3), col="green");
stat.info <- sprintf("\n\nmean: %.5g   median: %.5g   std: %.5g   range: [%.5g,%.5g]",
mean(sf, na.rm=TRUE), median(sf, na.rm=TRUE), sd(sf, na.rm=TRUE), min(sf), max(sf));
title(main=main, sub=stat.info);
dev.off();
pdf("avg_background.pdf");
pop <- "igr";
xlab <- "sample";
ylab <- "average background";
plot(avg.bg, main="", xlab=xlab, ylab=ylab, col="blue");
stat.info <- sprintf("\n\nmean: %.5g   median: %.5g   std: %.5g   range: [%.5g,%.5g]",
mean(avg.bg, na.rm=TRUE), median(avg.bg, na.rm=TRUE), sd(avg.bg, na.rm=TRUE), min(avg.bg), max(avg.bg));
title(main=main, sub=stat.info);
dev.off();
pdf("controls.pdf");
main <- sprintf("IGR Population - Relapse\n(%i patients on chip %s)",
as.integer(length(sampleNames(abatch))), annotation(abatch));
prs <- apply(spike == "P", 2, sum);
prs <- (prs/nbr.exp)*100;
sub <- sprintf("percent of present calls -> bioB: %i, bioC: %i, bioD: %i, creX: %i",
as.integer(prs[1]), as.integer(prs[3]), as.integer(prs[3]), as.integer(prs[4]));
```

```
xlab <- "sample";
ylab <- "housekeeping control";
plot(gapdh35, xlim=c(0, nbr.exp+30), ylim=c(0,4), main="", xlab=xlab, ylab=ylab, col="blue", pch="o");
points(gapdh3M, col="blue", pch="x");
points(beta.actin35, col="red", pch="o");
points(beta.actin3M, col="red", pch="x");
abline(h=1, col="black");
abline(h=3, col="green");
abline(h=1/3, col="green");
legend(x=nbr.exp, y=3.5, legend=c("GAPDH 3'/5'", "GAPDH 3'/M", "beta actin 3'/5'",
"beta actin 3'/M"), col=c("blue", "blue", "red", "red"), pch=c("o","x","o","x"));
title(main=main, sub=sub);
dev.off();
###############################################################################
#use of the AffyExtensions package
#image of weights computed for fitPLM
#boxplot of the fitPLM object
#Mbox of the fitPLM object
###############################################################################
#matrix of scatterplots
eset <- esetA.mas.igr;
maxp <- 25;
#generate names for the pairs files
nm <- NULL;
for (i in 1:ceiling(nbr.exp/maxp)) {
        nm <- c(nm, sprintf("pairs%i.bmp", as.integer(i)));
}
main <- sprintf("IGR Population - Relapse\n(%i patients on chip %s)",
as.integer(nbr.exp), annotation(eset));
for (i in 1:length(nm)) {
        if (maxp < nbr.exp) {
                if (i == length(nm)) {
                        ind <- (nbr.exp-maxp):nbr.exp;
                }
                else {
                        beg <- (i-1)*maxp;
                        end <- beg+maxp;
                        if (i == 1) {
                                beg <- 1;
                        }
                        ind <- beg:end;
                }
        }
        else {
                ind <- 1:nbr.exp;
        }
        bitmap(nm[i], height=100, width=100, res=150);
        pairs(log2(exprs(eset)[,ind]), pch=".", main="", cex.labels = 0.15, font.labels=0.15);
        title(main=main);
        dev.off();
}
###############################################################################
#matrix of MA plots
eset <- esetA.mas.igr;
maxp <- 10;
#generate names for the pairs files
nm <- NULL;
for (i in 1:ceiling(nbr.exp/maxp)) {
        nm <- c(nm, sprintf("mvapairs%i.bmp", as.integer(i)));
}
main <- sprintf("IGR Population - Relapse\n(%i patients on chip %s)",
as.integer(nbr.exp), annotation(eset));
for (i in 1:length(nm)) {
        if (maxp < nbr.exp) {
                if (i == length(nm)) {
                        ind <- (nbr.exp-maxp):nbr.exp;
                }
```

```
                else {
                        beg <- (i-1)*maxp;
                        end <- beg+maxp;
                        if (i == 1) {
                                beg <- 1;
                        }
                        ind <- beg:end;
                }
        }
        else {
                ind <- 1:nbr.exp;
        }
        bitmap(nm[i], height=100, width=100, res=150);
        #have to find good parameters for the fonts !!!
        mva.pairs(exprs(eset)[,ind], main="");
        title(main=main);
        dev.off();
}
```

Appendix F

Code of medic_read.R

```
#Function to read medic data
#
#init
if (!require(affy)) {
        stop("require affy library!");
}
medic <- function() {
#local functions
################################################################################
sn.fct <- function(lst) {
        return (lst[[1]]);
}
################################################################################
transform.name <- function(population, filename) {
        if (population=="oxford") {
                cut.sn <- "[^0-9]";
                fn <- filename;
                sn <- lapply(strsplit(fn, cut.sn), sn.fct);
                sn <- unlist(sn);
                pop <- rep("oxford", length(sn));
                fn <- fn[sort(as.numeric(sn), index.return=TRUE)$ix];
                sn <- sn[sort(as.numeric(sn), index.return=TRUE)$ix];
                #add the string "experiment_" before the experiment number to correspond with demographics file
                sn <- paste("experiment", sn, sep="_");
        }
        else if (population=="igr") {
                cut.sn <- " ";
                fn <- filename;
                sn <- lapply(strsplit(fn, cut.sn), sn.fct);
                sn <- unlist(sn);
                #remove the char 'P' at the beginning of the experiment name
                sn <- substr(sn, 2, nchar(sn));
                pop <- rep("igr", length(sn));
                fn <- fn[sort(as.numeric(sn), index.return=TRUE)$ix];
                sn <- sn[sort(as.numeric(sn), index.return=TRUE)$ix];
                #add the string "experiment_" before the experiment number to correspond with demographics file
                sn <- paste("experiment", sn, sep="_");
        }
        else if (population=="all") {
                cut.sn.oxford <- "[^0-9]";
                cut.sn.igr <- " ";
                fn <- filename;
                sn.oxford <- fn[substr(fn,1,1) != "P"];
                sn.igr <- fn[substr(fn,1,1) == "P"];
                sn.oxford <- lapply(strsplit(sn.oxford, cut.sn.oxford), sn.fct);
                sn.oxford <- unlist(sn.oxford);
                sn.igr <- lapply(strsplit(sn.igr, cut.sn.igr), sn.fct);
                sn.igr <- unlist(sn.igr);
                #remove the char 'P' at the beginning of the experiment name
```

```
                sn.igr <- substr(sn.igr, 2, nchar(sn.igr));
                pop <- c(rep("oxford", length(sn.oxford)), rep("igr", length(sn.igr)));
                sn <- c(sn.oxford, sn.igr);
                fn <- fn[sort(as.numeric(sn), index.return=TRUE)$ix];
                sn <- sn[sort(as.numeric(sn), index.return=TRUE)$ix];
                #add the string "experiment_" before the experiment number to correspond with demographics file
                sn <- paste("experiment", sn, sep="_");
        }
        else {
                stop("population name invalid!");
        }
        return (list(fn=fn, sn=sn, population=pop));
}
################################################################################
read.population <- function(population, my.wd, progn.file, nbr.experiments) {
                #read the prognosis information
                progn <- read.csv(file=progn.file, row.names=1);
                #keep the sammple names
                progn.sn <- row.names(progn);
                #keep the first column of data (relapse<5)
                progn <- progn[,1];

                #remove the experiment where there is no prognosis
                progn.sn <- progn.sn[!is.na(progn)]
                progn <- progn[!is.na(progn)]
                wd.save <- getwd();
                setwd(my.wd);
                #list the CEL files in the target directory
                fn <- dir(pattern="*.[CEL|cel]");
                if (length(fn) == 0) {
                        cat("no CEL file to read!\n");
                        return (list(err=-1));
                }
                fsn <- transform.name(population=population, filename=fn);
                fn <- fsn[[1]];
                sn <- fsn[[2]];
                pop <- fsn[[3]];
                #take only the experiments which names are in the prognosis file
                progn.checked <- vector(mode="numeric", length=length(sn));
                for (i in 1:length(sn)) {
                        if (is.element(sn[i], progn.sn)) {
                                progn.checked[i] <- progn[is.element(progn.sn, sn[i])];
                        }
                        else {
                                progn.checked[i] <- NA;
                        }
                }
                #remove the NA values
                sn <- sn[!is.na(progn.checked)];
                fn <- fn[!is.na(progn.checked)];
                pop <- pop[!is.na(progn.checked)];
                progn.checked <- progn.checked[!is.na(progn.checked)];
                if (nbr.experiments > 0 && nbr.experiments < length(fn)) {

                        fn <- fn[1:nbr.experiments];
                        sn <- sn[1:nbr.experiments];
                        progn.checked <- progn.checked[1:nbr.experiments];
                }
                #phenoData
                phdata <- new("phenoData", pData=as.data.frame(cbind(filename=fn, experimentname=sn,
prognosis=progn.checked, population=pop)), varLabels=list("filename", "experimentname", "prognosis", "population"))
                setwd(wd.save);

                return (list(filename=fn, phenodata=phdata));
}
################################################################################
```

```
        list (
                medic.read = function (population, my.wd, progn.file, nbr.experiments=0, verbose=TRUE) {
        #function to read CEL files
        #all these functions can be applied chip per chip to save memory (like just.rma()
        d.read <- read.population(population, my.wd, progn.file, nbr.experiments);
        fn <- d.read[[1]];
        phdata <- d.read[[2]];
        nbr.exp <- length(fn);
        if (nbr.experiments > 0 && nbr.experiments < nbr.exp) {
                fn <- fn[1:nbr.experiments];
                phdata <- phdata[1:nbr.experiments,];
        }
        print(pData(phdata));
     wd.save <- getwd();
        setwd(my.wd);

        ##########################
        #read the cel files in a affybatch object
        ##########################

        #read the first CEL files to get information about affymetrix chip
        eset <- ReadAffy(filenames=fn, phenoData=phdata, verbose=TRUE);
        setwd(wd.save);

        return (eset);
},
#############################################################################
                medic.read.mas = function (population, my.wd, progn.file, nbr.experiments=0, read.group=1, verbos
                        #function to read CEL files 1 by 1 and apply
                        #background correction: mas
                        #normalization: mas
                        #pm correction: mas
                        #summary: mas
        #all these functions can be applied chip per chip to save memory (like just.rma()

        d.read <- read.population(population, my.wd, progn.file, nbr.experiments);
        fn <- d.read[[1]];
        phdata <- d.read[[2]];
        nbr.exp <- length(fn);
        print(pData(phdata));
     wd.save <- getwd();
        setwd(my.wd);

        ##########################
        #construct the exprSet object with a modified expression measures matrix
        ##########################

        i <- 0;
        while (i < nbr.exp) {

                if ((i+read.group) > nbr.exp) {
                        j <- nbr.exp;
                }
                else {
                        j <- i+read.group;
                }
                if (i==0) {#first CEL files to read
                        #read the first CEL files to get information about affymetrix chip
                        eset <- ReadAffy(filenames=fn[(i+1):j], verbose=TRUE);
                        #use methods described above (chip per chip methods)
                        eset <- expresso(eset, bgcorrect.method="mas", normalize.method="constant",
pmcorrect.method="mas", summary.method="mas", verbose=TRUE);
                }
                else {

                        eseti <- ReadAffy(filenames=fn[(i+1):j], verbose=TRUE);
```

```
                        #use methods described above (chip per chip methods)
                        eseti <- expresso(eseti, bgcorrect.method="mas", normalize.method="constant",
pmcorrect.method="mas", summary.method="mas", verbose=TRUE);

                        #merge the current exprSet object and the old exprSet
                        slot(eset, "exprs") <- cbind(exprs(eset), exprs(eseti));
                        slot(eset, "se.exprs") <- cbind(se.exprs(eset), se.exprs(eseti));
                        #free memory
                        rm(eseti);
                }
                i <- j;
        }
        slot(eset, "phenoData") <- phdata;
        setwd(wd.save);

        return (eset);
},
###############################################################################
                medic.read.justrma = function (population, my.wd, progn.file, nbr.experiments=0) {

        d.read <- read.population(population, my.wd, progn.file, nbr.experiments);
        fn <- d.read[[1]];
        phdata <- d.read[[2]];
        print(pData(phdata));

        wd.save <- getwd();
        setwd(my.wd);
        #background correction using RMA background correction
        #data normalization using quantile normalization
        #the exprSet result contains the expression measures in log2 scale
        eset <- just.rma(filenames=fn, phenoData=phdata, notes="Read affy data with just.rma()
method because of memory limitations", verbose=TRUE, background=TRUE, normalize=TRUE);
        #unlog the expression measures
        cat("unlog expression measures\n");
        slot(eset, "exprs") <- 2^exprs(eset);
        setwd(wd.save);

        return (eset);
}
###############################################################################
        )
}
```

Appendix G

Code of medic.R

```
#functions used for the medic project
#method to treat duplicated genes between affymetrix HG-U133A an HG-U133B
action.duplic.gene <- function (eset, duplic.method="remove") {
        if (!require(Biobase)) {
                stop("require Biobase library!");
        }

        #mean all pairs of duplicated genes
        #warning: no management of NA values!
        #warning: no management of se.exprs values!

        exprs.temp <- exprs(eset);
        se.exprs.temp <- se.exprs(eset);
        dupl.gene <- geneNames(eset) [duplicated(geneNames(eset))];
        if (duplic.method == "mean") {
                for (i in 1:length(dupl.gene)) {
                        dupl.ix <- row.names(exprs.temp)==dupl.gene[i];
                        dupl <- sort(!dupl.ix, index.return=TRUE)$ix;
                        #indices of duplicated genes are in dupl[1] and dupl[2] if only 2 duplicated genes
                        exprs.temp[dupl[1],] <- apply(exprs(eset)[dupl.ix,], 2, mean);
                        exprs.temp <- exprs.temp[-dupl[2],];
                        se.exprs.temp <- se.exprs.temp[-dupl[2],];
                }
        }
        else if (duplic.method == "remove") {

                        dupl.ix <- !duplicated(geneNames(eset))
                        exprs.temp <- exprs(eset)[dupl.ix,];
                        se.exprs <- se.exprs(eset)[dupl.ix,];
        }
        else {
                cat("no valid method to treat the duplicated gene!\n");
        }
        cat(sprintf("%i duplicated genes\n", as.integer(length(dupl.gene))));
        slot(eset, "exprs") <- exprs.temp;
        slot(eset, "se.exprs") <- se.exprs.temp;
        return (eset);
}
#merge two exprSet objects
merge.exprset <- function(eset1, eset2) {
        if (!require(Biobase)) {
                stop("require Biobase library!");
        }
        if (length(setdiff(geneNames(eset1), geneNames(eset2))) == 0) {#merge experiments
                cat("merge by experiments\n");
                nbr.exp <- (length(experimentNames(eset1)) + length(experimentNames(eset2)));
                #phenoData
                phdata <- new("phenoData", pData=rbind(pData(eset1), pData(eset2)), varLabels=varLabels(eset1));
                #renumbering the experiments
```

```
                row.names(pData(phdata)) <- 1:nbr.exp;
                nexprs <- cbind(exprs(eset1), exprs(eset2));
                #renumbering the experiments
                dimnames(nexprs)[[2]] <- 1:nbr.exp;

                se.nexprs <- cbind(se.exprs(eset1), se.exprs(eset2));
                #renumbering the experiments
                dimnames(se.nexprs)[[2]] <- 1:nbr.exp;
                esetm <- new("exprSet", exprs=nexprs, se.exprs=se.nexprs,
phenoData=phdata, annotation=annotation(eset1), description=paste(description(eset1),
" and ", description(eset2)), notes=paste(notes(eset1), " and ", notes(eset2)));
        }
        else {#merge genes
                cat("merge by genes\n");
                esetm <- new("exprSet", exprs=rbind(exprs(eset1), exprs(eset2)),
se.exprs=rbind(se.exprs(eset1), se.exprs(eset2)), phenoData=phenoData(eset1),
annotation=paste(annotation(eset1), " and ", annotation(eset2)), description=paste(description(eset1),
" and ", description(eset2)), notes=paste(notes(eset1), " and ", notes(eset2)));
        }
        return (esetm);
}
#function to compute the intersection of a list
intersect.list <- function(list.input) {
        if (length(list.input) == 0) {
                cat("empty list!\n");
                return (-1);
        }
        res <- list.input[[1]];
        for (i in 2:length(list.input)) {

                res <- intersect(res, list.input[[i]]);
        }
        return (res);
}
```

Appendix H

Code of medic_simple_classif.R

```
medic.simple.classif <- function(eset, duplic.method="mean", classif.method="svm",
hyst1=15, hyst2=0.1, cl.thresh=0.5, nk=5, cost=1.5*10^-5, gamma=1.5*10^-5, clw=10) {
        ##########################
        #libraries
        ##########################

        if (!require(Biobase)) {
                stop("require Biobase library!");
        }
        if (classif.method == "svm") {
                        if (!require(e1071)) {
                                stop("require e1071 library!");
                        }
        }
        else if (classif.method == "knn") {
                if (!require(class)){
                        stop("require class library!");
                        }
        }
        else if (classif.method == "lm") {
                if (!require(base)){
                        stop("require base library!");
                        }
        }
        else {
                stop("classification method not valid!");
        }
###############################################################################
        ##########################
        #functions
        ##########################
action.duplic.gene <- function (eset, duplic.method="remove") {

        #mean all pairs of duplicated genes
        #warning: no management of NA values!
        #warning: no management of se.exprs values!

        exprs.temp <- exprs(eset);
        se.exprs.temp <- se.exprs(eset);
        dupl.gene <- geneNames(eset) [duplicated(geneNames(eset))];
        if (duplic.method == "mean") {
                for (i in 1:length(dupl.gene)) {
                        dupl.ix <- row.names(exprs.temp)==dupl.gene[i];
                        dupl <- sort(!dupl.ix, index.return=TRUE)$ix;
                        #indices of duplicated genes are in dupl[1] and dupl[2] if only 2 duplicated genes
                        exprs.temp[dupl[1],] <- apply(exprs(eset)[dupl.ix,], 2, mean);
                        exprs.temp <- exprs.temp[-dupl[2],];
                        se.exprs.temp <- se.exprs.temp[-dupl[2],];
                }
```

```
        }
        else if (duplic.method == "remove") {

                dupl.ix <- !duplicated(geneNames(eset))
                exprs.temp <- exprs(eset)[dupl.ix,];
                se.exprs <- se.exprs(eset)[dupl.ix,];
        }
        else {
                stop("no valid method to treat the duplicated gene!");
        }
        cat(sprintf("%i duplicated genes\n", as.integer(length(dupl.gene))));
        slot(eset, "exprs") <- exprs.temp;
        slot(eset, "se.exprs") <- se.exprs.temp;
        return (eset);
}
##############################################################################
        cat(sprintf("\nclassification by %s\n", classif.method));
        #verif for duplicated gene in the expression set
        if (sum(duplicated(geneNames(eset))) != 0) {

                cat("duplicated genes in the expression measures set\n");
                eset <- action.duplic.gene(eset=eset, duplic.method=duplic.method);
        }
        nbr.exp <- length(experimentNames(eset));
        nbr.gene <- length(geneNames(eset));
        cat(sprintf("\n%i genes in %i experiments\n", as.integer(nbr.gene), as.integer(nbr.exp)));
        #class weights
        if (classif.method == "svm") {
                #the class weights are also used in the evaluation criteria
                class.weights <- c(1, clw);
                names(class.weights) <- c("0", "1");
                regr.method <- FALSE;
        }
        if (classif.method == "knn") {
                regr.method <- FALSE;
        }
        if (classif.method == "lm") {
                #the class weights are also used in the evaluation criteria
                class.weights <- rep(1, nbr.exp);
                #give a weight of 10 for the relapses
                class.weights[pData(eset)[,3] == 1] <- clw;
                regr.method <- TRUE;
        }
        #format the output
        res.gm <- data.frame(matrix(data=0, nrow=nbr.exp, ncol=(nbr.gene)),
row.names=as.character(pData(eset)[,2]));
        names(res.gm) <- geneNames(eset);
        res.miscl.global <- NULL;
        res.miscl <- data.frame(matrix(data=0, nrow=nbr.exp, ncol=2),
row.names=as.character(pData(eset)[,2]));
        names(res.miscl) <- c("false_positive", "false_negative");
        #save the error rate plots in the feature selection
        pdf("feature_selection.pdf");
        #misclassification (false positive, false negative)
        misclassif.global <- c(0,0);
        for (k in 1:nbr.exp) {

                cat(sprintf("\nstep %i:\n", as.integer(k)));
                #split the expression set for the leave-one-out
                group.loo <- rep(1, nbr.exp);
                group.loo[k] <- 2;
                eset.split <- split(eset, group.loo);
                eset.in <- eset.split[[1]];
                eset.out <- eset.split[[2]];
                rm(list="eset.split");
                #eset.in contains the data for the training set
                #eset.out contains the data for the validation set
```

```
#rank the genes according to their correlation with the outcome
#pearson correlation coefficient

gcor <- cor(x=t(exprs(eset.in)), y=pData(eset.in)[,3], method="pearson");
gene.ranked <- geneNames(eset.in)[sort(abs(gcor), index.return=TRUE)$ix];
gcor <- gcor[sort(abs(gcor), index.return=TRUE)$ix];
nexp <- length(experimentNames(eset.in));
#for the feature selection, the internal cross-validation estimation of svm is used
loo.internal <- c(nexp,clw*nexp);
#control the improvement evolution during feature selection
hyst <- clw*floor(nexp / hyst1);        #class weight * 20% of the number of experiments
hyst.param <- clw*hyst2;
cat(sprintf("feature selection -> max of %i '\#' will be displayed:\n", as.integer(hyst)));
nbr.rel <- sum(pData(eset.in)[,3] == 1);
nbr.notrel <- sum(pData(eset.in)[,3] == 0);
miscl.save <- c(nbr.notrel, nbr.rel);

for (i in 2:nbr.gene) {#feature selection
        #X contains all the data to build the model
        X <- as.data.frame(t(exprs(eset.in)[gene.ranked[1:i],]))
        #bug when only 1 marker gene ...
        Y <- as.data.frame(pData(eset.in)[,3]);
        if (classif.method == "lm") {#lm does not support factors
                Y[,1] <- as.numeric(Y[,1])-1;
        }
        X <- cbind(Y, X);
        row.names(X) <- pData(eset.in)[,2];
        names(X) <- c("prognosis", gene.ranked[1:i]);
        #leave-one-out validation for feature selection
        miscl <- c(0,0);
        for (h in 1:nrow(X)) {
                if (classif.method == "svm") {
                        m <- svm(prognosis ~ ., data=X[-h,], class.weights=class.weights);
                }
                if (classif.method == "lm") {
                        m <- lm(prognosis ~ ., data=X[-h,], weights=class.weights);
                }
                #no training require for knn
                #evaluate this set on the left experiment
                if (classif.method == "svm") {
                        cl <- predict(m, X[h,-1]);
                }
                if (classif.method == "lm") {
                        cl <- predict(m, X[h,-1]);
                }
                if (classif.method == "knn") {
                        cl <- knn(train=X[-h,-1], test=X[h,-1], cl=X[-h,1], k = nk);
                }
                if (regr.method) {
                        if (cl < cl.thresh) {
                                cl <- 0;
                        }
                        else {
                                cl <- 1;
                        }
                }

                if (cl != X[h,1]) {
                        if (cl == 1 && X[h,1] == 0) {#false positives
                                miscl[1] <- miscl[1] + 1;
                        }
                        else {#false negatives
                                miscl[2] <- miscl[2] + 1;
                        }
                }
        }
```

```
                        #save the misclassification rate
                        miscl.save <- rbind(miscl.save, miscl);
                        #decrease the threshold for the hysteresis
                        if (i%%hyst.param == 0 && hyst >0) {
                                hyst <- hyst-1;
                                cat("\#");
                        }
                        if ((loo.internal[1]+(clw*loo.internal[2])) < (miscl[1]+(clw*miscl[2]))-hyst && i >= 10) {
                                break;
                        }
                        if ((loo.internal[1]+(clw*loo.internal[2])) >
(miscl[1]+(clw*miscl[2]))) {#fp + clw * fn = misclassification rate
                                loo.internal <- miscl;
                                best.gmarker <- gene.ranked[1:i];
                        }
                }
                #best.gmarker contains the best set of marker genes
                cat("\n");
                cat(sprintf("%i selected marker genes - misclassificatIon -> (%i,%i)\n",
as.integer(length(best.gmarker)), as.integer(loo.internal[1]), as.integer(loo.internal[2])));
                #save plots for feature selection
                main <- sprintf("Feature Selection Misclassification Rate\nL
eave-One-Out Step %i", as.integer(k));
                xlab <- "number of genes";
                ylab <- "percent of misclassification rate";
                plot(miscl.save[,1]/nbr.notrel, col="blue", type="l", ylim=c(0,1),
xlim=c(0,nrow(miscl.save)+30), xlab=xlab, ylab=ylab, main=main);
                lines(miscl.save[,2]/nbr.rel, col="red");
                abline(v=length(best.gmarker), col="green");
                legend(x=nrow(miscl.save)+1, y=0.6, legend=c("% false positive", "% false negative",
"\# marker genes"), col=c("blue", "red", "green"));
               #save the best marker genes
                res.gm[k,is.element(names(res.gm), best.gmarker)] <- 1;
                res.miscl[k,] <- miscl;
                #keep only the marker genes selected by feature selection
                X <- X[,is.element(names(X), c("prognosis", best.gmarker))];
                if (classif.method == "svm") {
                        #fit the svm
                        m <- svm(prognosis ~ ., data=X, class.weights=class.weights);

                        #evaluate this set on the left experiment
                        cl <- predict(m, t(exprs(eset.out)[is.element(geneNames(eset.out), best.gmarker),]));
                }
                if (classif.method == "knn") {
                        cl <- knn(train=X[,-1], test=t(exprs(eset.out)[is.element(geneNames(eset.out),
best.gmarker),]), cl=X[,1], k = nk);
                }
                if (classif.method == "lm") {
                        #bug
                }
                if (cl != pData(eset.out)[,3]) {
                        cat("bad classification\n");
                        if (cl == 1 && pData(eset.out)[,3] == 0) {#false positives
                                misclassif.global[1] <- misclassif.global[1] + 1;
                        }
                        else {#false negatives
                                misclassif.global[2] <- misclassif.global[2] + 1;
                        }
                }
                else {
                        cat("good classification\n");
                }
                print(misclassif.global);
        }
        dev.off();
```

```
        res.miscl.global <- misclassif.global;
        return (list(misclassification=res.miscl, marker.gene=res.gm,
global.misclassification=res.miscl.global));
}
```

Appendix I

Code of medic_struct_id.R

```
#structural identification for the medic simple analysis
medic.struct.id <- function(eset, classif.method="svm", clw=10, marker.gene=100,
cost=c(-15, 15), gamma=c(-15, 5), nk=c(3,30)) {
        ##########################
        #libraries
        ##########################

        if (!require(Biobase)) {
                stop("require Biobase library!");
        }
        if (classif.method == "svm") {
                        if (!require(e1071)) {
                                stop("require e1071 library!");
                        }
        }
        else if (classif.method == "knn") {
                if (!require(class)){
                        stop("require class library!");
                        }
        }
        else if (classif.method == "lm") {
                if (!require(base)){
                        stop("require base library!");
                        }
        }
        else {
                stop("classification method not valid!");
        }
################################################################################
        ##########################
        #functions
        ##########################
action.duplic.gene <- function (eset, duplic.method="remove") {

        #mean all pairs of duplicated genes
        #warning: no management of NA values!
        #warning: no management of se.exprs values!

        exprs.temp <- exprs(eset);
        se.exprs.temp <- se.exprs(eset);
        dupl.gene <- geneNames(eset) [duplicated(geneNames(eset))];
        if (duplic.method == "mean") {
                for (i in 1:length(dupl.gene)) {
                        dupl.ix <- row.names(exprs.temp)==dupl.gene[i];
                        dupl <- sort(!dupl.ix, index.return=TRUE)$ix;
                        #indices of duplicated genes are in dupl[1] and dupl[2] if only 2 duplicated genes
                        exprs.temp[dupl[1],] <- apply(exprs(eset)[dupl.ix,], 2, mean);
                        exprs.temp <- exprs.temp[-dupl[2],];
                        se.exprs.temp <- se.exprs.temp[-dupl[2],];
```

```
                }
        }
        else if (duplic.method == "remove") {

                dupl.ix <- !duplicated(geneNames(eset))
                exprs.temp <- exprs(eset)[dupl.ix,];
                se.exprs <- se.exprs(eset)[dupl.ix,];
        }
        else {
                stop("no valid method to treat the duplicated gene!");
        }
        cat(sprintf("%i duplicated genes\n", as.integer(length(dupl.gene))));
        slot(eset, "exprs") <- exprs.temp;
        slot(eset, "se.exprs") <- se.exprs.temp;
        return (eset);
}
################################################################################
cat(sprintf("\nstructural identification for %s classification\n", classif.method));
        #verif for duplicated gene in the expression set
        if (sum(duplicated(geneNames(eset))) != 0) {

                cat("duplicated genes in the expression measures set\n");
                eset <- action.duplic.gene(eset=eset, duplic.method=duplic.method);
        }
        nbr.exp <- length(experimentNames(eset));
        nbr.gene <- length(geneNames(eset));
        cat(sprintf("\n%i genes in %i experiments\n", as.integer(nbr.gene), as.integer(nbr.exp)));
        #class weights
        if (classif.method == "svm") {
                #the class weights are also used in the evaluation criteria
                class.weights <- c(1, clw);
                names(class.weights) <- c("0", "1");
                regr.method <- FALSE;
        }
        if (classif.method == "knn") {
                regr.method <- FALSE;
        }
        if (classif.method == "lm") {
                #the class weights are also used in the evaluation criteria
                class.weights <- rep(1, nbr.exp);
                #give a weight of 10 for the relapses
                class.weights[pData(eset)[,3] == 1] <- clw;
                regr.method <- TRUE;
        }
        #save the error rate plots in the feature selection
        pdf("structural_identification.pdf");
        #rank the genes according to their correlation with the outcome
        #pearson correlation coefficient

        gcor <- cor(x=t(exprs(eset)), y=pData(eset)[,3], method="pearson");
        gene.ranked <- geneNames(eset)[sort(abs(gcor), index.return=TRUE)$ix];
        gcor <- gcor[sort(abs(gcor), index.return=TRUE)$ix];
        #take only the first marker.gene genes
        gene.ranked <- gene.ranked[1:marker.gene];
        gcor <- gcor[1:marker.gene];
        if (classif.method == "svm") {
                #X contains all the data to build the model
                X <- as.data.frame(t(exprs(eset)[gene.ranked,]))
                Y <- as.data.frame(pData(eset)[,3]);
                X <- cbind(Y, X);
                row.names(X) <- pData(eset)[,2];
                names(X) <- c("prognosis", gene.ranked);
                #first grid search (find a good region of parameters space)
                cat("\nlarge grid search:\n");
                cost <- 2^(seq(cost[1], cost[2], 2));
                gamma <- 2^(seq(gamma[1], gamma[2], 2));
```

```
                tuned.param <- tune.svm (prognosis ~ ., data=X, gamma=gamma, cost=cost,
class.weights=class.weights);
                print(tuned.param[1:2]);
                #second grid search (refine a good region of parameters space)
                cat("fine grid search:\n");
                g2 <- as.integer(log2(tuned.param$best.parameters[1]));
                c2 <- as.integer(log2(tuned.param$best.parameters[2]));

                cost <- 2^(seq((c2-1), (c2+1), 0.25));
                gamma <- 2^(seq((g2-1), (g2+1), 0.25));
                tuned.param <- tune.svm(prognosis ~ ., data=X, gamma=gamma, cost=cost,
class.weights=class.weights);
                print(tuned.param[1:2]);
                cat("\n");
                cost <- tuned.param$best.parameters[1];
                gamma <- tuned.param$best.parameters[2];
                res.param <- list(cost=cost, gamma=gamma);
                #plot of performances
                xlab <- "cost";
                ylab <- "gamma";
                zlab <- "misclassification rate";
                main <- sprintf("Structural Identification for %s classification", classif.method);
                sub <- "grid search";
                scatterplot3d(x=tuned.param$performances[,1], y=tuned.param$performances[,2],
z=tuned.param$performances[,3], xlab=xlab, ylab=ylab, zlab=zlab, main=main, sub=sub, pch="+");
        }

        if (classif.method == "knn") {
                #X contains all the data to build the model
                X <- as.data.frame(t(exprs(eset)[gene.ranked,]))
                Y <- as.data.frame(pData(eset)[,3]);
                X <- cbind(Y, X);
                row.names(X) <- pData(eset)[,2];
                names(X) <- c("prognosis", gene.ranked);

                tuned.param <- tune.knn(x=X[,-1], y=X[,1], k = nk[1]:nk[2]);
                print(tuned.param[1:2]);
                cat("\n");
                #nk <- tuned.param$;
                res.param <- list(k=nk);

                #plot of performances
                xlab <- "k";
                ylab <- "misclassification rate";
                main <- sprintf("Structural Identification for %s classification", classif.method);
                sub <- "exhaustive search";
                plot(x=tuned.param$performances[,1], y=tuned.param$performances[,2], xlab=xlab,
ylab=ylab, main=main, sub=sub, type="l", col="red");
                abline(v=tuned.param$best.parameters, col="green");
        }

        if (classif.method == "knn1") {
                for (i in nk[1]:nk[2]) {

                        miscl <- c(0,0);
                        for (h in 1:nrow(X)) {

                                #evaluate this set on the left experiment
                                cl <- knn(train=X[-h,-1], test=X[h,-1], cl=X[-h,1], k = i);

                                if (cl != X[h,1]) {
                                        if (cl == 1 && X[h,1] == 0) {#false positives
                                                miscl[1] <- miscl[1] + 1;
                                        }
                                        else {#false negatives
                                                miscl[2] <- miscl[2] + 1;
                                        }
```

```
                                }
                        }
                        #save the misclassification rate
                        miscl.save <- rbind(miscl.save, miscl);
                        #decrease the threshold for the hysteresis
                        if (i%%hyst.param == 0 && hyst >0) {
                                hyst <- hyst-1;
                                cat("\#");
                        }
                        if ((loo.internal[1]+(clw*loo.internal[2])) < (miscl[1]+(clw*miscl[2]))-hyst && i >= 10) {
                                break;
                        }
                        if ((loo.internal[1]+(clw*loo.internal[2])) > (miscl[1]+(clw*miscl[2]))) {#fp + clw * fn =
                                loo.internal <- miscl;
                                best.gmarker <- gene.ranked[1:i];
                        }
                }
        }
        dev.off();
        return (res.param);
}
```