Statistical Analysis of Genotype and Gene Expression Data

Dissertation

by

Holger Schwender

Submitted to
the Department of Statistics
of the University of Dortmund

in Fulfillment of
the Requirements for the Degree of
Doktor der Naturwissenschaften

February 22, 2007

Referees:

Prof. Dr. Katja Ickstadt

Prof. Dr. Claus Weihs

Date of Oral Examination: February 20, 2007

Abstract

A common and important goal in cancer research is the identification of genetic markers such as genes or genetic variations that enable to determine if a person has a particular type of cancer, or lead to a higher risk of developing cancer.

In recent years, many biotechnologies for measuring these markers have been developed. The most prominent examples are microarrays that can be used to, e.g, measure the expression levels of tens of thousands of genes simultaneously.

The most widely used type of microarrays is the Affymetrix GeneChip on which each gene is represented by eleven pairs of probes. The corresponding probe intensities have to be preprocessed, i.e. summarized to one expression value per gene, before variable selection and classification methods can be applied to the gene expression data.

This thesis is based on two projects: The goals of the first project are to identify the preprocessing method for Affymetrix microarrays that leads to the most efficient data reduction, and to provide a software enabling to apply this procedure to the data from studies comprising hundreds of Affymetrix GeneChips. The results of this project are presented in this thesis.

The second project is concerned with SNPs (Single Nucleotide Polymorphisms), i.e. variations at a single base-pair position in the genome. While a vast number of papers on the analysis of gene expression data have been published, only a few variable selection and classification methods dealing with the specific needs of the analysis of SNP data have been proposed. One of the exceptions is logic regression. In this thesis, it is shown how approaches for the analysis of gene expression data can be adapted to SNP data, and a procedure based on a bagging version of logic regression is proposed that enables the detection of SNP interactions explanatory for a higher cancer risk. Furthermore, two measures for quantifying the importance of each of these interactions for prediction are presented, and compared with existing measures.

Acknowledgement

First and foremost, I would like to thank my advisor, Katja Ickstadt, for giving me the opportunity to perform research in her group, and the freedom to explore and solve many problems myself, while always supporting my research with very helpful discussions. I could not have asked for a better advisor, and I am looking forward to continuing working with her.

This thesis is based on two projects – one in cooperation with Roche Diagnostics, Penzberg, and the other at the Collaborative Research Center 475 ("Reduction of Complexity in Multivariate Data Structures") of the Department of Statistics, University of Dortmund. I therefore would like to gratefully acknowledge the financial support of both Roche Diagnostics and the Deutsche Forschungsgemeinschaft (DFG).

Moreover, I would like to thank the GENICA research network for allowing us to analyze their genotype (and environmental) data, and in particular, Christina Justenhoven, IkP Stuttgart, for answering several questions about biotechnologies.

I am deeply grateful to Anton Belousov and Friedemann Krause, Roche Diagnostics, for countless fruitful discussions, and for sharing their wisdom with me.

I would like to thank Klaus Jung, MPC Bochum, for proofreading this thesis, and for several interesting discussions.

Furthermore, I would like to thank Uwe Ligges, Department of Statistics, University of Dortmund, for giving me advice on numerous problems concerned with `R` and the building of packages in `R`, and Ingo Ruczinski, Johns Hopkins

University, Baltimore, MD, for providing me the `R` code for the CART based imputation method, and for giving me an insight into some of the technical details of logic regression.

Finally, a very special thanks goes to Andreas Krause, Pharsight Corporation. I am not sure if I would have ever got involved in the highly interesting field of statistical analysis of microarrays if he would not have given an excellent talk about microarrays at the Department of Statistics in 2002, and me the opportunity to write a diploma thesis about this topic.

Contents

PART I

Background Information

Chapter 1

Introduction

Advances in biotechnology have enabled the usage of genetic markers such as genes and genetic variations in studies concerned with the detection of causes for complex diseases. In particular in cancer research, these markers are employed to identify, on the one hand, genes allowing to determine if a person has a particular (sub-)type of cancer, and on the other hand, genetic variations such as single nucleotide polymorphisms (SNPs) leading to a higher risk of developing cancer.

A SNP is a single base-pair position in the DNA sequence at which (typically two) different base alternatives exist that each occur in at least 1% of a population. Since the human genome is diploid, i.e. consists of pairs of chromosomes, each SNP is explained by two bases – one from each chromosome. Thus, each SNP can take three values/genotypes: A SNP is of the homozygous reference (or the homozygous variant) genotype if both chromosomes show the more (or the less) frequent base. If one of the bases is the less, and the other the more frequent variant, then the SNP is of the heterozygous variant genotype.

SNPs and the expression levels of genes can be measured in a similar way using one of several biotechnologies. The two most prominent types of such methods are microarrays (e.g., Brown and Botstein, 1999, Lipshutz et al., 1999) and polymerase chain reaction (PCR; see, e.g., Strachan and Read, 2005). While the latter enables to determine the expression level of a gene – which is a mea-

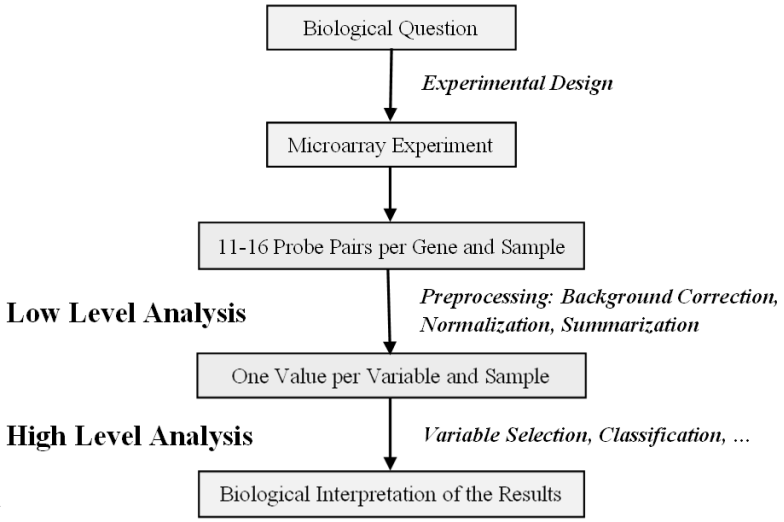

FIGURE 1.1. Role of Statistics in Affymetrix Microarray Experiments.

surement of the activity of this gene – or the genotype of a SNP accurately, microarrays can be employed to quantify the expression levels of tens of thousands of genes, or the genotypes of hundreds of thousands of SNPs simultaneously.

The most widely used type of DNA microarrays is the Affymetrix GeneChip on which each gene is represented by typically eleven pairs of oligonucleotide probes, i.e. short mRNA sequences consisting of 25 bases. Before procedures for high level analyses such as variable selection and classification can be applied to the expression values, these signals thus have to be generated from the probe intensities provided by the microarrays (see Figure 1.1 for a diagram of this proceeding). Since these intensities are a result of a trade-off of quality for quantity, they are perturbed by noise arising from, e.g., the way of measuring. Hence, procedures are needed that are able, on the one hand, to remove not only this noise but also technical effects such as laboratory or operator effects, and on the other hand, to reduce the 22 probe intensities per gene and sample/microarray to one expression value such that the results of the subsequent high level analyses are still meaningful.

Typically, procedures for this preprocessing or low level analysis consist of three steps: In the first step, the probe intensities are corrected for global back-

ground noise. In the second step, the probe intensities are normalized – which should remove technical effects. Finally, the background-corrected and normalized intensities are summarized to one signal per gene and sample.

Not all preprocessing methods follow exactly this scheme. For example, in MAS 5.0 (**M**icro**A**rray **S**uite **5.0**; Affymetrix, 2002), the standard Affymetrix algorithms until July 2004, the normalization is performed after the summarization such that not the probe intensities but the expression values are normalized.

A study of Roche Diagnostics, Penzberg, is concerned with the comparison of MAS 5.0 with, on the one hand, in-house modifications of the current standard Affymetrix algorithm PLIER (**P**robe **L**ogarithmic **I**ntensity **ER**ror estimation; Affymetrix, 2005) that are called PLA and PLA+ (**P**lier **L**ike **A**lgorithm; Liu, 2004) throughout this thesis, and on the other hand, RMA (**R**obust **M**ulti-array **A**verage; Irizarry et al., 2003), the most popular academic alternative to the Affymetrix algorithms, and PLM (**P**robe **L**evel **M**odel; Bolstad, 2004) that only differs from RMA in the summarization step. The main goal of this study is the identification of the algorithm leading to the best data reduction, and hence, the specification of a procedure that will be employed as the standard preprocessing method in upcoming Affymetrix microarray projects of Roche Diagnostics.

All but the internal Roche methods can be applied to the probe data generated in a DNA microarray experiment using functions freely available at the web page of BioConductor (http://www.bioconductor.org; Gentleman et al., 2004), an open source and open development project for the analysis of genetic data in the statistical software environment `R` (Ihaka and Gentleman, 1996).

A drawback of `R` is that one runs quickly into massive memory problems if the data set is high-dimensional. For example, generating the RMA signals in a study comprising several ten Affymetrix HG-U133_Plus_2 chips, the most widely used type of Affymetrix DNA microarrays, is only possible with a large amount of RAM. Employing the standard BioConductor functions, other more memory-intensive procedures such as PLM cannot be applied in such a study.

However, in a project of Roche Diagnostic concerned with colorectal cancer

(CRCA), more than 400 HG-U133_Plus_2 chips have to be preprocessed using a machine with 4 GB of RAM on which Windows XP is installed. Therefore, a second goal of this study is to develop a strategy for preprocessing such a huge number of Affymetrix microarrays on this computer using `R`.

The results of this study are presented in Part II of this thesis. The set of preprocessing procedures considered in the comparison is, however, extended by adding, on the one hand, PLIER with and without normalization, and on the other hand, versions of RMA and PLM in with the standard background correction approach is replaced by a method that takes the base composition of the probe sequences into account which is assumed to improve the estimation of the RMA and PLM signals (cf. Wu et al., 2004).

While Part II is concerned with low level analysis of gene expression data, Part III addresses high level analysis of SNP data and is based on the project "Statistical Complexity Reduction in Molecular Epidemiology" of the Collaborative Research Center 475 at the University of Dortmund. The major goal of this project is the development of methods for the identification of polymorphisms, interactions of polymorphisms, and interactions of polymorphisms and epidemiological variables that lead to a higher risk of developing cancer. Since in this project the analysis of the data set from the GENICA (interdisciplinary study group on **G**ene **EN**vironment **I**nteraction and breast **CA**ncer in Germany) study is of particular interest, it serves as the main example in the applications presented in Part III. (For more details on the GENICA study, see Appendix A.2, or http://www.genica.de.)

In this thesis, we focus on the most common type of polymorphisms, i.e. on SNPs. We are thus interested in the detection of SNPs and SNP interactions showing a distribution that substantially differs between several groups, and the construction of a classification rule based on such features.

While in recent years a huge number of papers on high level analyses of gene expression data have been published, only a few variable selection and classification methods have been proposed for the analysis of SNP data (e.g., Cordell

and Clayton, 2002, Ritchie et al., 2001, Ruczinski et al., 2003; for an overview of procedures that might be applied to SNP data for variable selection, see Heidema et al., 2006). However, many of the procedures for analyzing continuous gene expression data can be adapted to categorical SNP data. To exemplify this, we show in this thesis how three popular methods particularly developed for the analysis of DNA microarray data can be modified for SNPs.

Since missing values are a common problem in association studies such as the GENICA study (Dai et al., 2006), a method proposed by Troyanskaya et al. (2003) for imputing missing expression values based on k Nearest Neighbors (kNN; Fix and Hodges, 1951) is considered as a first example. Afterwards, it is shown how the Significance Analysis of Microarrays (SAM; Tusher et al., 2001), a multiple testing procedure that utilizes a QQ plot to adjust for multiplicity, can be applied to SNP data. Finally, a discrimination method called Prediction Analysis of Microarrays (PAM; Tibshirani et al., 2002) that can cope with a vast number of continuous variables is adapted to categorical data.

A problem in the analysis of high-dimensional data is that, e.g., the distances between tens of thousands of pairs of observations or variables have to be computed (when, e.g., using kNN), or several thousand test statistics have to be calculated several hundred times (when employing a permutation method such as SAM). This can lead to very long run times in `R`, in particular, if these statistics are determined one by one.

A solution to this problem is to parallelize the computation by, e.g., employing matrix algebra. Therefore, for each of the three examples, an algorithm is presented that makes essential use of matrix calculations, and reduces the run time substantially.

Since not individual SNPs but interactions of SNPs are assumed to be responsible for complex diseases such as sporadic breast cancer (Garte, 2001, Culverhouse et al., 2002), SAM and PAM are not only applied to the SNPs themselves, but it is also shown how they can be used to analyze interactions of SNPs.

One of the major goals in the analysis of genotype data is the construction

of classification rules such as

> "If SNP S_1 is of the heterozygous variant genotype AND
> SNP S_2 is of the homozygous variant genotype OR both SNP
> S_3 AND S_4 are NOT of the homozygous reference genotype,
> then a person has (a higher risk to develop) a particular disease."

A procedure developed for solving exactly this type of problems is logic regression proposed by Ruczinski et al. (2003). This adaptive regression and classification methodology attempts to identify Boolean combinations of binary variables for predicting, e.g., the case-control status of an observation.

Other discrimination methods such as CART (Breiman et al., 1984), bagging (Breiman, 1996), Random Forests (Breiman, 2001), and Support Vector Machines (SVMs; Vapnik, 2000) can also be applied to SNP data (Schwender et al., 2004). But in comparisons, on the one hand, with CART and Random Forests (Ruczinski et al., 2004), and on the other hand, with other regression procedures (Kooperberg et al., 2001, Witte and Fijal, 2001), logic regression has shown a good performance in the application to SNP data.

We therefore consider logic regression more closely. As a starting point, it is determined if logic regression also outperforms PAM and the other above-mentioned discrimination methods when applied to the genotype data sets examined in this thesis (see Appendix A).

Afterwards, a procedure based on a bagging version of logic regression is introduced that enables the identification of SNPs and SNP interactions associated with the covariate of interest. Furthermore, two measures for quantifying the importance of each of the interactions detected by this approach called logicFS are proposed. The advantage of these approaches over existing quantities such as the variable importance measures of CART and Random Forests, or the squared weights used in RFE-SVM (**R**ecursive **F**eature **E**limination using **S**upport **V**ector **M**achines; Guyon et al., 2002) is that the importances of not only the variables themselves but also the interactions can be determined

without including the interactions as variables in the discrimination procedure.

The main parts of this thesis, Part II and Part III, are based on two different projects: One project on the low level analysis of Affymetrix DNA microarray data, and the other on high level analysis of SNP data. However, gene expression data can, of course, also be employed to, e.g., construct a classification rule. As mentioned above, this is, in fact, the actual goal of many of the microarray experiments. Contrariwise, SNPs can be genotyped using microarrays. Therefore, we shortly explain, on the one hand, how gene expression data might be employed for variable selection and discrimination (Chapter 6; in particular, Section 6.1), and on the other hand, how genotypes of SNPs can be measured (Section 2.2.2) and preprocessed (Section 3.7) using Affymetrix microarrays.

This thesis is organized as follows: In Chapter 2, background information on genetics and on the biotechnologies used to generate the real gene expression and genotype data described in Appendix A is given. While the preprocessing methods summarized in Chapter 3 are compared in Chapter 4, it is shown in Chapter 5 how PLM can be applied to data from experiments comprising a huge number of microarrays. In Chapter 6, the adaption of the three DNA microarray methods to SNP data is presented, whereas Chapter 7 contains the comparison of logic regression with other discrimination procedures. In Chapter 8, logicFS and the two importance measures are introduced and compared with a similar approach based on logic regression. Furthermore, a method required by logicFS for converting a logic expression into a disjunctive normal form, i.e. an OR-combination of AND-combinations, is presented. Finally, the results of the analyses are summarized and discussed in Chapter 9. In the Appendix, detailed information on the data sets used in the analyses are given, supplementary plots and tables are displayed, `R` packages containing functions for procedures introduced in this thesis are presented, and statistical methods mainly employed in Part II are shortly described.

Chapter 2

Genetic and Biotechnological Background

2.1 Genetic Background

What is gene expression? And what are SNPs? In this section which is a modified excerpt from Schwender et al. (2006b), these and other important genetic terminologies and concepts are explained.

A more detailed introduction to genetics is given, e.g., by Alberts et al. (2005), or by Gonick and Wheelis (1991).

2.1.1 The Human Genome

Everybody is composed of zillions of *cells*. Virtually any of these cells comprises the complete *human genome* in its *nucleus*. The human genome consisting of 23 pairs of *chromosomes* is the blueprint for all cellular structures and activities in the human body. In each of these pairs, one chromosome comes from the mother, and the other from the father. Each chromosome is a huge chain of two intertwined strands of *deoxyribonucleic acid* (*DNA*), the *double-helix*. As shown in Figure 2.1, each DNA strand is a long sequence of *nucleotides*, where each nucleotide is a molecule consisting of a phosphate group, a deoxyribose sugar,

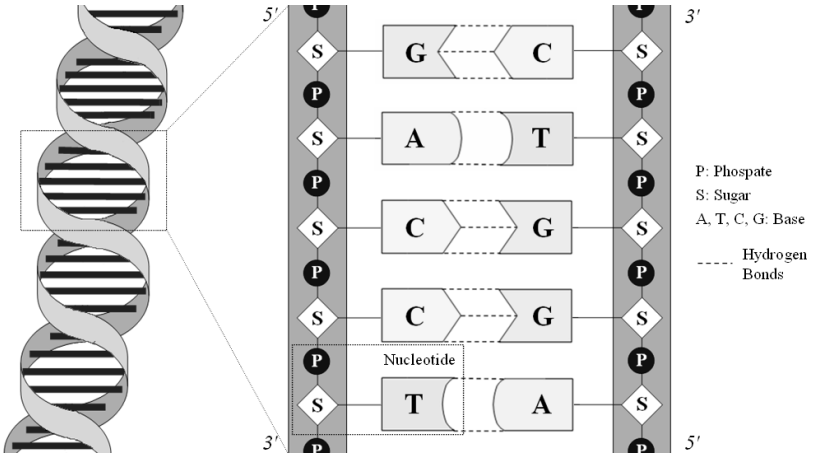

FIGURE 2.1. The DNA.

and one of the four bases *adenine* (A), *thymine* (T), *cytosine* (C) and *guanine* (G).

Even though there are two strands, one sequence consisting of the letters A, T, C and G suffices to describe the DNA because of the *complementary base-pairing*: A on one of the strand is always connected via *hydrogen bounds* to T on the other strand, whereas C is always paired with its complement G. Thus, if we know the sequence of one of the strands, we also know the sequence of the other strand. The leading end of each of these strands is called *5' end*, and the tail end *3' end*. Since they are complementary, one strand runs from 5' to 3', and the other from 3' to 5'.

Only small segments of the DNA, namely the *genes*, contain construction information for *proteins*. Since proteins are responsible for the structure and the activity of a cell, and hence, for virtually everything that happens in an organism, it is important to understand how genes are translated into proteins. The *Central Dogma of Molecular Biology* displayed in Figure 2.2 gives an answer to this question:

Starting at the 5' end, the information in the genes is first *transcribed* into single-stranded *messenger ribonucleic acid* (*mRNA*) by a process based on the abovementioned complementary base-pairing. RNA is similar to DNA, but car-

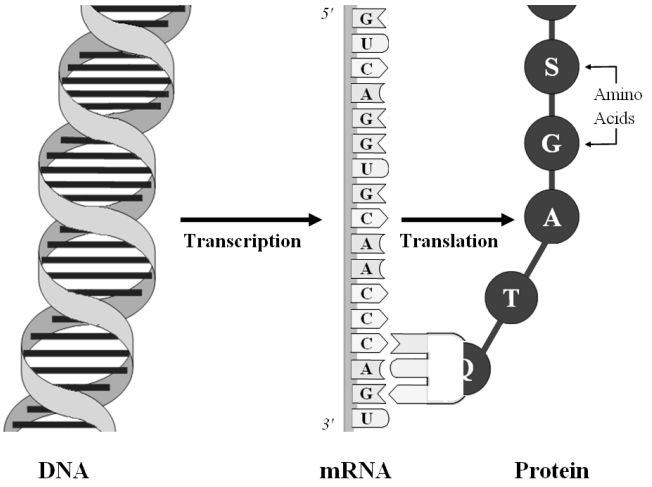

FIGURE 2.2. Central Dogma of the Molecular Biology.

ries a different sugar molecule (ribose instead of deoxyribose) and a different base (*uracil* (U) instead of thymine).

After the transcription, the mRNA leaves the nucleus of the cell. Starting again at the 5' end of the mRNA, each *codon*, i.e. each triplet of nucleotides, is *translated* into one of twenty *amino acids*. (Note that some of the triplets code for the same amino acid, since there are actually $4^3 = 64$ possible triplets.) Finally, all amino acids corresponding to one mRNA sequence form a chain that folds into a protein.

The relation between the codons and the amino acids is known as the *genetic code*. While *gene expression* actually denotes the process of converting the DNA sequence of a gene into a protein, in the analysis of microarrays the abundance of specific mRNA in a sample is referred to as (*gene*) *expression level*.

A gene, however, does not always code for the same protein. Since only small parts of a gene, namely *exons*, are needed in the translation step, *introns*, i.e. non-coding regions of a gene, are removed in the translation step by a process called *RNA-splicing*. Since not always the same exons are retained, different combinations of exons can be spliced to produce different *mRNA isoforms* of a gene that can lead to different proteins. It is assumed that about 60% of the genes are affected by this *alternative splicing*.

But this is not the only reason that a gene does not always code for the same protein. DNA itself varies between humans. Even though humans share far more than 99% of their DNA, there are still millions of base pair positions at which the DNA can differ. Such variations in the DNA sequence of a gene might also lead to different proteins.

2.1.2 Genetic Variations

There are several forms of genetic variation, ranging from deletions or substitutions of single or multiple bases over translocations of large segments of a chromosome to changes in the number of chromosomes.

Such *mutations* cannot only affect the physical appearance of an individual but also the development of a (complex) disease. If a change in just one of the two chromosomes building a pair is sufficient to alter the *phenotype*, i.e. an observable characteristic of an organism, then the mutation is called *dominant*. If both chromosomes must be affected by this variation to change the phenotype, it is called *recessive*.

Each of several forms a DNA sequence can take is called *allele*. If a specific locus in the DNA sequence is considered, then the allele that occurs less often in the population of interest is referred to as *minor allele*. If the frequency of this allele is larger than 1%, the variation is called *polymorphism*. (This condition is necessary to distinguish inherited variations from spontaneous mutations.)

The most common type of polymorphisms are *SNPs* (***S**ingle **N**ucleotide **P**olymorphisms*) that are characterized by the possibility of different bases at a specific base-pair position. Furthermore, a deletion or insertion at a particular locus is also referred to as SNP.

Since the human genome is *diploid*, i.e. consists of pairs of chromosomes, not just one DNA sequence but both chromosomes are typically considered in *association studies*. Thus, the *genotype*, i.e. the combination of the two alleles – one from each chromosome – is analyzed in such studies.

Since typically two sequence alternatives exist at a specific base-pair position, a SNP can take three genotypes: The SNP is of the *homozygous reference* (or the *homozygous variant*) *genotype*, if both bases are the more (or the less) frequent variant. The SNP is of the *heterozygous variant genotype*, if one of the bases is the less frequent, and the other is the more frequent variant, where it is not possible to specify which chromosome shows which of the two bases.

2.2 Biotechnological Background

Now we know what gene expression and SNPs are. But how can these genetic variables be measured? In this section, an answer to this question is given. In Section 2.2.1, the idea behind the Affymetrix GeneChip is explained, and it is described how this biotechnology can be employed to measure expression levels.

Since the SNPs from the GENICA study have been genotyped using a combination of PCR (**P**olymerase **C**hain **R**eaction) and MALDI-TOF-MS (**M**atrix **A**ssisted **L**aser **D**esorption/**I**onization – **T**ime **O**f **F**light – **M**ass **S**pectrometry), Section 2.2.2 focusses on these two biotechnologies, and contains only a short description of how Affymetrix microarrays can be used to measure SNPs.

While full details on the Affymetrix GeneChip technology can be found in Affymetrix (2001, 2003), Kennedy et al. (2003) describe how this biotechnology can be employed to genotype SNPs. Introductions to other types of microarrays such as cDNA chips or BeadArrays are given, e.g, by Brown and Botstein (1999) or by Kuhn et al. (2004), respectively. PCR is explained in more details, e.g., by Strachan and Read (2005), whereas MALDI-TOF-MS is described by Pusch et al. (2002).

2.2.1 Measuring Gene Expression Data

Proteins are responsible for virtually anything that happens in an organism – e.g., the development of cancer. It would therefore be interesting to measure the abundance of proteins in cancer cells, and to compare them with normal,

i.e. non-cancer, cells. Even though methods for monitoring protein expressions such as 2D gels (Klose and Kobalz, 1995) and protein microarrays (Sydor and Nock, 2003) already exist, it remains very complex to measure a large number of proteins simultaneously.

Following the Central Dogma of Molecular Biology, the function of cells can, however, also be investigated by monitoring mRNA levels if the assumption that most of the mRNA is translated into proteins holds.

Tens of thousands of such mRNA levels can be measured simultaneously using *DNA microarrays* such as the Affymetrix GeneChip that consists of a 1.28 cm x 1.28 cm glass slide, or *array*, comprising hundreds of thousands of *probe cells*, i.e. locations, on the slide. For example, the Affymetrix HG-U133_Plus_2 chip is composed of $1,164 \times 1,164 = 1,354,896$ probe cells. Virtually any of these cells contains millions of copies of a specific *oligonucleotide probe* (or short: *oligo*), i.e. an mRNA sequence consisting of 25 bases.

Each gene is represented by at least one *probe set* typically composed of eleven *Perfect Match* (*PM*) and eleven *Mismatch* (*MM*) oligos. But not any probe set corresponds to a gene. The Affymetrix HG-U133_Plus_2 chip, e.g., comprises 54,675 probe sets which either represent one of about 38,000 genes, or are employed for quality control (for more information on control probe sets, see Section 4.3).

As illustrated by Figure 2.3, a PM oligo is a *25mer* complementary to a part of the mRNA sequence of interest that should be unique for the target gene, i.e. the mRNA sequence of none of the other genes should contain this sequence.

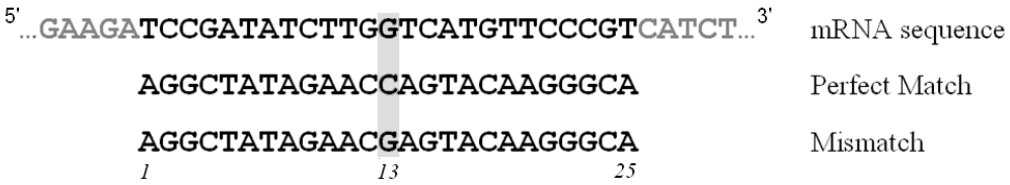

FIGURE 2.3. Perfect Match and Mismatch. (*Source*: Schwender and Belousov, 2006)

If single-stranded RNA is *hybridized*, i.e. joined, to the probes on the chip by the process described below, the RNA of a gene represented by a particular PM is assumed to bind to this PM. However, not only the RNA of interest will in general hybridize to the intended PM, but also RNA that is not supposed to join with this PM. To adjust for this non-specific binding and for background noise caused by the way the arrays are prepared and measured, each chip also contains a mismatch oligo for each PM. As shown in Figure 2.3, the sequence of a MM – that builds a *probe pair* with the corresponding PM – is identical to sequence of this PM except for the 13^{th} base which is complementary to the 13^{th} base of the PM.

If the expression levels of the genes in a tissue sample should be measured *total RNA*, i.e. a mixture of mRNA and two other types of RNA (*ribosomal RNA* and *transfer RNA*) that do not code for proteins, is isolated from this tissue, and then reverse-transcribed to produce double-stranded *complementary DNA* (*cDNA*). This cDNA serves as a template in the subsequent *in vitro transcription* (*IVT*) reaction in which (single-stranded) *complementary RNA* (*cRNA*) is multiplied using PCR (see Section 2.2.2) and labeled with fluorescent dye. This cRNA is fragmented into pieces typically consisting of 25-200 bases and given onto the chip. After 16 hours of *hybridization*, the non-binding cRNA is removed from the chip, and the array is scanned to measure the amount of fluorescence at each probe cell by a 16-bit image, where the amount of fluorescence in a particular probe cell is assumed to be proportional to the abundance of the specific mRNA.

Depending on the used Affymetrix GeneChip technology, each probe cell is represented by up to $8 \times 8 = 64$ pixel values, where each of these pixels can take a value between 0 and $2^{16} - 1$ (since a 16-bit image is used). These values are summarized to one intensity per probe cell by removing the border pixels and computing the 75% quantile of the remaining inner pixels. For each microarray, the resulting probe intensities are stored in a *CEL file*, the starting point of the methods presented in Chapter 3.

2.2.2 Genotyping SNPs

Affymetrix GeneChips can also be employed to genotype SNPs. As Figure 2.4 shows, each SNP is not represented by a set of probe pairs, but by a set composed of typically ten *probe quartets* each consisting of one PM and one MM for each sequence alternative.

FIGURE 2.4. Probe quartet representing the SNP marked by the red background. (The gray background marks the 13$^{\text{th}}$ base which differs between PM and MM.)

In association studies such as the GENICA study in which several tens of SNPs are considered, SNPs are typically genotyped using a combination of PCR with another molecular-biological technique such as MALDI-TOF-MS.

PCR which has enabled the boom in molecular genetics since the early 1980s is a technology to *amplify*, i.e. multiply, a specific DNA sequence to get enough genetic material such that it can be analyzed with standard biochemical procedures. In each of the typically 20-30 cycles of a PCR process, the amount of DNA is doubled such that the original amount of the DNA sequence is amplified by a factor of 2^{20}-2^{30}. Each cycle consists of the following three steps:

1. *Denaturation*: The DNA double strand is separated by heating it to a temperature of 94°-96° C.

2. *Annealing*: The temperature is lowered slowly to 50°-60° C so that two specific *primers*, i.e. short oligos matching the beginning of the two single-stranded DNA sequences produced during the denaturation, are able to anneal to the respective single-stranded DNA sequence, and thus to mark the beginning of the DNA target sequence.

3. *Extension*: The four types of nucleotides and a *DNA polymerase* – a special

enzyme that catalyzes the synthesis of DNA – are added to the mixture. This reagent is heated to 72° C enabling the synthesis of two copies of the target sequence.

After amplifying the DNA region of interest, a short primer is annealed to this sequence, where the 3' end of the primer is located directly before the base-pair position of the SNP that should be genotyped. This primer is then extended allele-specificly by one or two bases. Since the molecular masses of the four nucleotides are known, the allele-specific products generated in the primer extension reaction can be determined in MALDI-TOF-MS by measuring their mass, and the genotype of the SNP of interest can be specified by the abundance of these products.

For this genotyping, the analyte comprising the extension products is embedded into a *matrix* consisting of crystallized molecules. Afterwards, this solution is placed into a *mass spectronometer*. A pulsed laser is used to evaporate both the matrix and the analyte, where the major fraction of the laser energy is absorbed by the matrix such that the analyte is transferred gently into the gas phase. Furthermore, the molecules of the analyte are ionized by a proton transfer from the matrix. (This soft ionization method called *MALDI* is required by the MS procedure, since otherwise the molecules of the analyte would decay immediately during the ionization/desorption.) These ions are accelerated by an electric field, and enter the field-free flight tube of the mass spectronometer.

Depending on their mass, the molecules reach the detector at the other end of the mass spectronometer to different time points. Therefore, the *time of flight* t from the evaporation to the arrival at the detector can be used as a measure for their molecular mass. Based on this time of flight, the *mass-to-charge ratio*

$$\frac{m}{z} = \frac{2 \cdot e \cdot U}{s^2} \cdot t^2$$

is computed, where $e = 1.602 \cdot 10^{-19}$ coulomb is the elementary charge, U is the acceleration voltage, and s is the length of the flight tube.

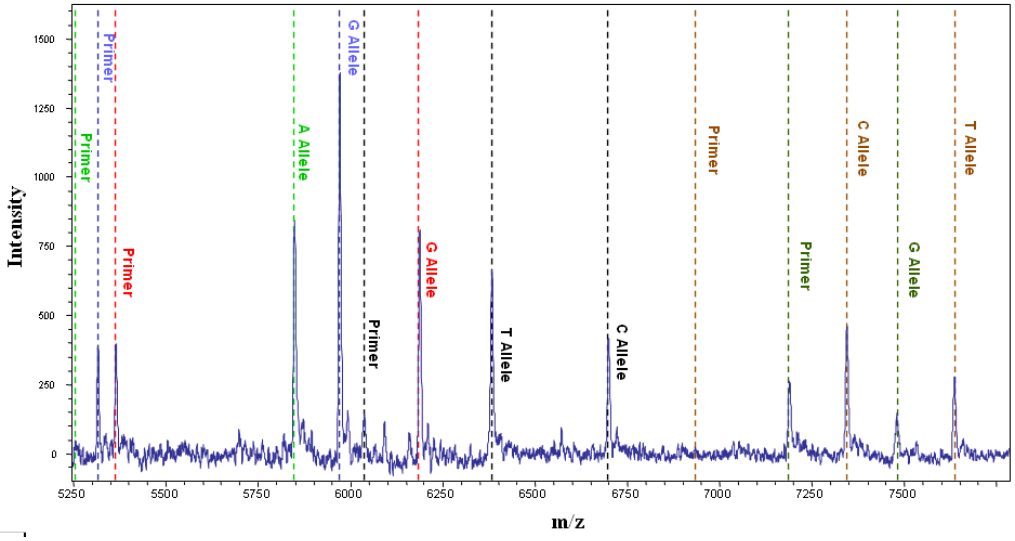

FIGURE 2.5. Mass spectrum of the multiplex reaction of 6 SNPs. Lines labeled by "Primer" specify the m/z-ratio of the non-extended primers. (Slightly modified version of a figure provided by Christina Justenhoven, IkP Stuttgart.)

The detector of the mass spectronometer not only identifies the extension products by their molecular mass, but also determines the abundances of the ions with the respective m/z-ratio. Finally, the genotype of the SNP is specified depending on the abundances of the allele-specific products.

In Figure 2.5, the mass spectrum of the simultaneous analysis of six SNPs is displayed. Since for the SNP marked by the light-green lines (on the left-hand side of Figure 2.5), only the A allele is abundant, and for the SNP represented by red lines, just the G allele shows a large intensity, both SNPs are homozygous, and exhibit an A or a G, respectively, at the corresponding base-pair positions. By contrast, the SNP marked by black lines is heterozygous, as both alleles are abundant.

2.3 Gene Expression vs. SNP Data

There are two major differences between gene expression and SNP data that have to be considered when analyzing these types of data. Firstly, expression

values are continuous, whereas SNPs are categorical variables with typically three categories. Secondly, while the expression of a gene depends, e.g., on the cell type and can change over time, the genotype of a SNP is always the same. Therefore, a time series analysis of SNPs does not make sense, whereas in DNA microarray experiments, it is a highly interesting topic (e.g., Wichert et al., 2004) which is, however, not part of this thesis. Another consequence of the constant nature of SNPs is that a person who does not show a particular cancer during an association study concerned with this type of cancer but formerly suffered from it should not be included as control in this study.

PART II

Preprocessing of Affymetrix GeneChips

Chapter 3

Preprocessing Methods

3.1 Introduction

An immediate goal in a study concerned with Affymetrix microarrays is to preprocess the probe data comprised by this study, i.e. to reduce the intensities of the typically eleven probe pairs per probe set to one expression value such that the results of further analyses such as feature selection and classification can still be meaningful.

As mentioned in Section 2.2.1, the PMs measure both the relative abundance of the respective gene and the amount of non-specific binding and background noise, whereas the MMs are intended to measure non-specific binding and background noise. Therefore, a first idea for computing the expression value of a gene would be to subtract the MM intensities from the corresponding PM values, and to average over the differences of the probe pairs representing this gene.

This approach, however, only works in an ideal Affymetrix world. As noted by Irizarry et al. (2003), in reality about 30% of the MMs are larger than the corresponding PMs.

A solution to this MM > PM riddle is given by Naef and Magnasco (2003). In an experiment comprising 86 Affymetrix HG-U95A arrays, they observe that the 13^{th} base of the PM sequences of 95% of the probe pairs in which the MM is larger than the PM is a purine, i.e. either A or G. Following Naef and

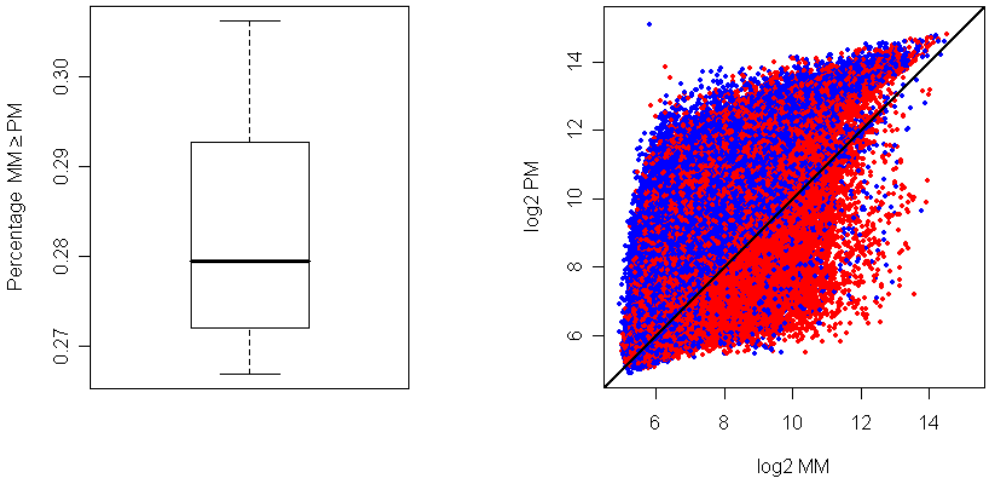

FIGURE 3.1. Left Panel: Box plot of the percentage of probe pairs consisting of a PM showing a smaller intensity than the corresponding MM for each of the 38 Affymetrix HG-U133_Plus_2 chips (see Appendix A.1). Right Panel: Scatter plot of the probe pairs of two of these microarrays. If the 13^{th} base of the PM is a purine, the pair is shown in red. Otherwise, it is displayed in blue.

Magnasco (2003), this behavior is based on the labeling process in which only the pyrimidines, i.e. C and U, are labeled with fluorescent dye, and on the fact that purines are larger molecules than pyrimidines.

Figure 3.1 reveals that in the analysis of the 38 Affymetrix HG-U133_Plus_2 arrays used in the comparison presented in Chapter 4 also about 28% of the MMs are larger than the corresponding PMs. However, only 62.8% of these PMs exhibit a purine as 13^{th} base. (Note that more than 60% of the probe pairs showing $\log_2$-transformed intensities less than 7.) Thus, there seem to be other factors that also cause the MMs to be larger than the PMs.

In Section 3.4.2, a preprocessing method based on the findings of Naef and Magnasco (2003) is presented. Other solutions to the MM > PM problem are, e.g., to totally ignore the MMs, or to compute idealized MMs that are always smaller than the corresponding PMs.

Many procedures have been developed that try to combat this and other problems of preprocessing. Usually, these methods consist of three steps: First, the probe intensities are corrected for global background noise, then they are

normalized, and finally summarized gene-wisely to expression values.

In the following sections, the former and the current standard algorithm of Affymetrix, MAS 5.0 and PLIER, the most popular academic alternative to these approaches, RMA, and some of its modifications, and two internal Roche Diagnostics methods called PLA and PLA+ are introduced. Afterwards, it is briefly described how RMA can be adapted to SNP microarrays.

In these sections, the PM and the MM belonging to the h^{th} probe pair, $h = 1, \ldots, H_i$, of the i^{th} probe set, $i = 1, \ldots, m$, on the j^{th} chip, $j = 1, \ldots, n$, are denoted by $PM_{hj}^{(i)}$ and $MM_{hj}^{(i)}$, respectively, where typically $H_i = 11$, m is in the tens of thousands, and n is in the tens. If all probe intensities comprised by a microarray are considered together, then the probe intensity at coordinate (x, y) of the chip, $x, y = 1, \ldots, \sqrt{n_{\text{chip}}}$, are denoted by p_{xy}, where n_{chip} is the number of probe cells on the array. We, however, abstain from using different notations for PMs and MMs (or in general probe intensities) that are in different stages of the preprocessing, since, on the one hand, the described approaches can be applied to the PMs and MMs no matter whether they have previously been background corrected and/or normalized, and on the other hand, adding another superscript to the PMs and MMs does not contribute to a better understanding of the preprocessing methods.

3.2 MicroArray Suite 5.0

After recognizing that their method of just taking the average over the probe pair differences to compute the expression values has several drawbacks (e.g., negative expression values, very noisy for low intensities), Affymetrix (2002) proposed a new algorithm called MAS 5.0 (**M**icro**A**rray **S**uite **5.0**).

In MAS 5.0, each probe intensity is background corrected by dividing the microarray into typically 16 zones (see Figure 3.2), and subtracting a probe specific background value depending on a weighted average over the 16 zone specific background values from the probe intensity. More details on this approach are

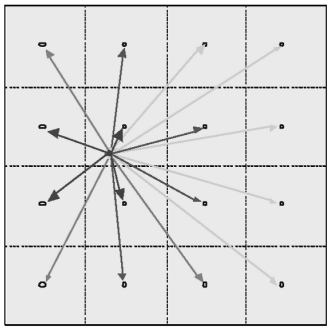

FIGURE 3.2. Schematic representation of the background correction procedure of MAS 5.0. (*Source*: Affymetrix, 2002, p. 4, slightly modified)

given in Algorithm 3.1.

Algorithm 3.1 (Background Correction of MAS 5.0)

1. Divide the array into 16 squares, and compute for each square k, $k = 1, \ldots, 16$, the region specific background value b_k and the zone specific noise s_k by the mean and the standard deviation, respectively, of the 2% lowest probe intensities in this square.

2. For each probe intensity p_{xy}, $x, y = 1, \ldots, \sqrt{n_{\text{chip}}}$,

   (a) compute the weights $w_k^{-1}(x, y) = (x - x_k)^2 + (y - y_k)^2 + 100$, where x_k and y_k are the coordinates of the centroid of region k, $k = 1, \ldots, 16$,

   (b) determine the probe specific background value and noise by

   $$b_{xy} = w^{-1}(x, y) \sum_{k=1}^{16} w_k(x, y) b_k \quad \text{and} \quad s_{xy} = w^{-1}(x, y) \sum_{k=1}^{16} w_k(x, y) s_k,$$

   where $w(x, y) = \sum_{k=1}^{16} w_k(x, y)$,

   (c) background correct p_{xy} by setting them to

   $$p_{xy}^{\text{new}} = \max\Big\{\max\big\{p_{xy}, 0.5\big\} - b_{xy},\, 0.5 s_{xy}\Big\}.$$

Afterwards, the background corrected PMs and MMs of probe set i, $i = 1, \ldots, m$, and sample j, $j = 1, \ldots, n$, are summarized to one expression value x_{ij} by firstly generating the ideal mismatches

$$IM_{hj}^{(i)} = \begin{cases} MM_{hj}^{(i)}, & \text{if } MM_{hj}^{(i)} < PM_{hj}^{(i)} \\ PM_{hj}^{(i)} \cdot 2^{-SB_{ij}}, & \text{if } MM_{hj}^{(i)} \geq PM_{hj}^{(i)} \text{ and } SB_{ij} > 0.03 \\ PM_{hj}^{(i)} \cdot 2^{-0.03/(1+(0.03-SB_{ij})/10)}, & \text{if } MM_{hj}^{(i)} \geq PM_{hj}^{(i)} \text{ and } SB_{ij} \leq 0.03 \end{cases},$$

$h = 1, \ldots, H_i$, where the probe set specific background SB_{ij} is determined by Tukey's one step biweight estimate T_{B} (Tukey, 1977) of the mean of $\log_2 PM_{hj}^{(i)} - \log_2 MM_{hj}^{(i)}$, $h = 1, \ldots, H_i$ (see Algorithm 3.2 with $\upsilon = 5$ and $\epsilon = 0.0001$). Secondly, x_{ij} is computed by

$$x_{ij} = T_{\mathrm{B}}\left(\log_2\left(PM_{hj}^{(i)} - IM_{hj}^{(i)}\right),\ h = 1, \ldots, H_i\right).$$

Since the IMs are always smaller than the corresponding PMs, all expression values are strictly positive.

Algorithm 3.2 (Tukey's One-Step Biweight Estimate)

Let $\{z_1, \ldots, z_H\}$ be a set of H observations, and υ and ϵ be given constant.

1. Denote the median and the MAD of $z_1, \ldots, z_H$ by M and D.

2. For $h = 1, \ldots, H$, compute

$$u_h = \frac{z_h - M}{\upsilon D + \epsilon} \quad \text{and} \quad w(u_h) = \left(1 - u_h^2\right)^2 \cdot I\left(\left|u_h\right| \leq 1\right).$$

3. Robustly estimate the mean of $z_1, \ldots, z_H$ by

$$T_{\mathrm{B}}\left(z_h,\ h = 1, \ldots, H\right) = \frac{\sum_{h=1}^{H} w(u_h) z_h}{\sum_{h=1}^{H} w(u_h)}$$

Finally, the expression values are normalized by scaling them array-wise such that the 2% trimmed mean of the values from each chip is 500. The expression values $x_{1j}, \ldots, x_{mj}$ of sample j, $j = 1, \ldots, n$, are thus normalized by setting them to

$$x_{ij}^{\text{new}} = \frac{500 \cdot q_{0.96}}{\sum_{i=q_{0.02}}^{m+1-q_{0.02}} x_{(i)j}} x_{ij}, \tag{3.1}$$

where $x_{(i)j}$ are the sorted expression values such that $x_{(1)j} \leq x_{(2)j} \leq \ldots \leq x_{(m)j}$, and $q_\alpha = \lfloor \alpha m \rfloor + 1$ is the largest integer smaller than or equal to $\alpha m + 1$.

3.3 Robust Multi-Array Average

Since MMs not only measure non-specific binding and background noise, but also contain information about the gene abundance intendedly probed by PMs, they actually should be considered in preprocessing methods. Irizarry et al. (2003), however, decided not to include MMs in their preprocessing procedure called RMA (**R**obust **M**ulti-array **A**nalysis), since when developing RMA they did not know how to extract this information.

Motivated by noticing that the density of the observed PM intensities typically look like the ones displayed in Figure 3.3, Irizarry et al. (2003) model the observed PM values of each sample j, $j = 1, \ldots, n$, as a sum of the specific

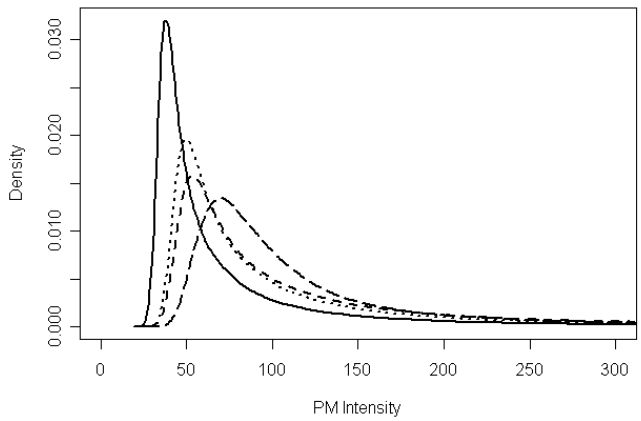

FIGURE 3.3. Density of the PM intensities of four of the 38 Affymetrix HG-U133_Plus_2 chips described in Appendix A.1.

signals $S \sim \text{Exp}(\gamma_j)$ and the background noise $N \sim \text{N}\left(\nu_j, \sigma_j^2\right)$, where S and N are assumed to be independent, and N is truncated at zero to avoid negative background corrected PM intensities.

Starting from this model, each PM intensity is background corrected by setting it to the expected signal $\text{E}\left(S \,\middle|\, O = PM_{hj}^{(i)}\right)$, where

$$\text{E}\big(S\big|\, O = o\big) = a_j + \sigma_j \frac{\phi\left(\frac{a_j}{\sigma_j}\right) - \phi\left(\frac{o-a_j}{\sigma_j}\right)}{\Phi\left(\frac{a_j}{\sigma_j}\right) + \Phi\left(\frac{o-a_j}{\sigma_j}\right) - 1} \tag{3.2}$$

with $a_j = o - \nu_j - \sigma_j^2\gamma_j$, and ϕ and Φ being the density and the distribution function, respectively, of the standard normal distribution (for a derivation of (3.2), see Bolstad, 2004).

In the actual implementation of RMA in the `R` function `rma`, $\Phi\left(\frac{o-a_j}{\sigma_j}\right) - 1$ and $\phi\left(\frac{o-a_j}{\sigma_j}\right)$ are omitted, since following Bolstad (2004) the latter value is negligible and $\Phi\left(\frac{o-a_j}{\sigma_j}\right) \approx 1$ in most microarray experiments. The parameters ν_j, γ_j and σ_j, $j = 1, \ldots, n$, are estimated by ad-hoc approaches: For each sample j, ν_j is estimated by the mode of the density of the PM intensities, σ_j is determined by the variability in the probe intensities less than $\hat{\nu}_j$, and γ_j is estimated by the reciprocal of the mode of the density of the strictly positive $\left(PM_{hj}^{(i)} - \hat{\nu}_j\right)$ values.

In a comparison of several normalization methods, Bolstad et al. (2003) identify quantile normalization described in Algorithm 3.3 as the approach that shows the best performance in terms of variance and bias reduction. Furthermore, the run time of quantile normalization is extremely short in comparison to the run times of other complete data methods, i.e. approaches that combine information from all arrays for normalization, that otherwise work almost as well as quantile normalization. The MA plots (see Appendix D.1) in Figure 3.4 reveal another important advantage of quantile normalization. Contrary to scaling, see (3.1), it can effectively combat non-linearities between arrays that are frequently observed in microarray experiments.

Therefore, the background corrected PMs are quantile normalized in the

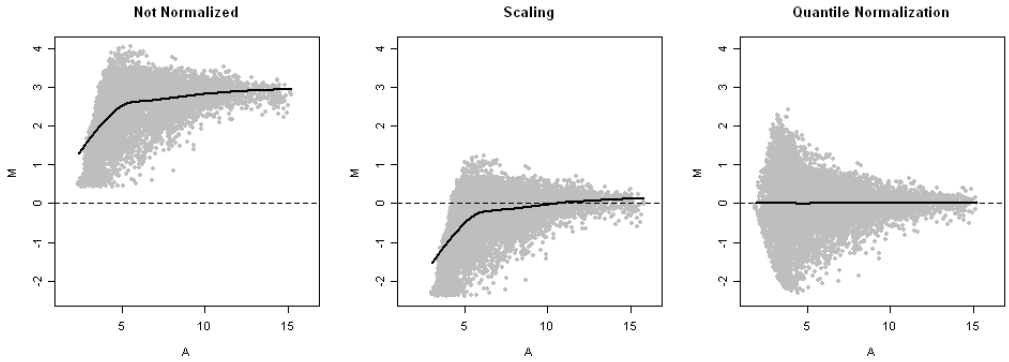

FIGURE 3.4. Scaling vs. Quantile Normalization. For a subset of the ($\log_2$-transformed) probe intensities of two of the 38 HG-U133_Plus_2 chips (see Appendix A.1), MA plots before normalization (left) and after normalization using scaling (middle) and quantile normalization (right) are shown. The solid lines are loess curves fitted through the data points. (*Source*: Schwender and Belousov, 2006)

normalization step of RMA by constructing a $\sum_{i=1}^{m} H_i \times n$ matrix in which each row corresponds to one of the $n_{\text{PM}} = \sum_{i=1}^{m} H_i$ PMs, and each row to one of the n arrays, and by applying Algorithm 3.3 to this matrix.

Algorithm 3.3 (Quantile Normalization)

Let $\mathbf{Z}$ be a $K \times n$ matrix.

1. Construct a $K \times n$ matrix $\mathbf{Z}^{\text{sort}}$ with elements $z_{kj}^{\text{sort}} = z_{(k)j}$, $k = 1, \ldots, K$, $j = 1, \ldots, n$.

2. Construct a $K \times n$ matrix $\mathbf{Z}^{\text{rank}}$ with entries $z_{kj}^{\text{rank}} = \left\lfloor \frac{1}{|\mathcal{T}_{kj}|} \sum_{\ell \in \mathcal{T}_{kj}} \ell \right\rfloor$ with $\mathcal{T}_{kj} = \left\{ \ell : z_{\ell j}^{\text{sort}} = z_{kj} \right\}$.

3. Set $\mathbf{q} = n^{-1}\mathbf{Z}^{\text{sort}}\mathbf{1}_n$, where $\mathbf{1}_n$ is a vector of length n containing only ones.

4. Normalize the columns of $\mathbf{Z}$ by setting $\mathbf{Z}$ to $\mathbf{Z}^{\text{new}}$ with elements

$$z_{kj}^{\text{new}} = \mathbf{q}_{z_{kj}^{\text{rank}}}, \quad k = 1, \ldots, K, \; j = 1, \ldots, n.$$

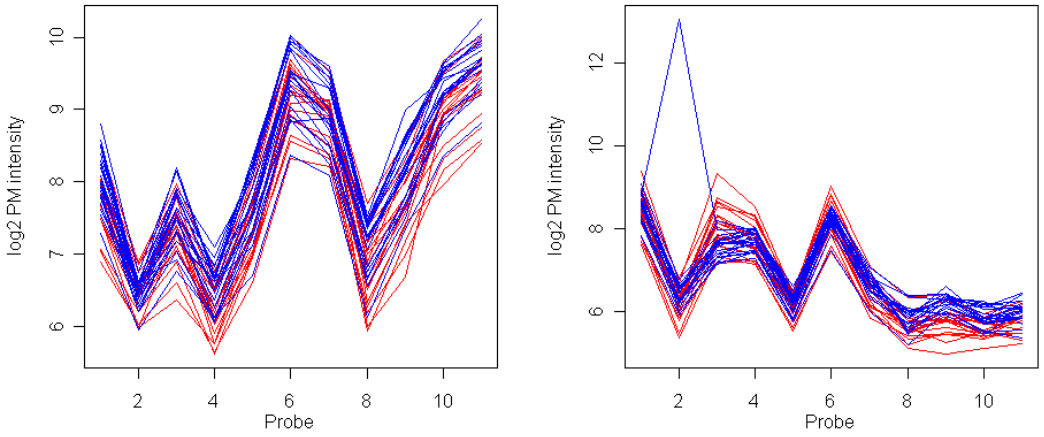

FIGURE 3.5. Profiles of the probe sets with Affymetrix-ID 242059_at (left panel) and 1563090_at (right panel) for the 18 colorectal (red) and the 20 breast (blue) cancer samples (see Appendix A.1).

As exemplified by the profiles of the probe sets displayed in Figure 3.5, the variability of a single probe across several chips is typically smaller than the variation in a probe set from a single array. The summarization of the intensities of a probe set might thus benefit from considering all samples in one model. In fact, Bolstad (2004) shows that fitting such a multi-chip model outperforms methods that examine each array separately by, e.g., computing the expression value of each probe set by a (robust) mean over the corresponding intensities (as in Section 3.2). As the right panel of Figure 3.5 reveals, occasionally occurring outliers are a problem for the summarization step. Since tens of thousands of multi-chip models should be fitted, a well-suited summarization method should therefore be able to deal with such outliers automatically. Assuming that the probe and the chip effects are multiplicative on the original scale, Irizarry et al. (2003) hence employ median polish (Tukey, 1977) described in Algorithm 3.4 to robustly fit a multi-chip model

$$\log_2 PM_{hj}^{(i)} = \mu^{(i)} + \alpha_h^{(i)} + \beta_j^{(i)} + \varepsilon_{hj}^{(i)}$$

for each probe set i, $i = 1, \ldots, m$, where $\mu^{(i)}$ is the intercept, $\alpha_h^{(i)}$ is the effect of probe h, $h = 1, \ldots, H_i$, and $\beta_j^{(i)}$ is the effect of chip j, $j = 1, \ldots, n$.

Algorithm 3.4 (Median Polish)

Let y_{hj}, $h = 1, \ldots, H$, $j = 1, \ldots, n$ be a set of observations, and τ be the tolerance for convergence.

1. Construct a $H \times n$ matrix $\mathbf{E}$ with initial entry $e_{hj} = y_{hj}$, and set $s_{\text{old}} = 0$.
2. Compute the vector $\mathbf{r}$ consisting of the row-wise medians of $\mathbf{E}$, and sweep $\mathbf{E}$ by setting its elements e_{hj} to $e_{hj}^{\text{new}} = e_{hj} - r_h$.
3. Generate the vector $\mathbf{c}$ consisting of the column-wise medians of $\mathbf{E}$, and update $\mathbf{E}$ by setting its entries e_{hj} to $e_{hj}^{\text{new}} = e_{hj} - c_j$.
4. Set $s_{\text{new}} = \sum_{h,j} |e_{hj}|$. If $s_{\text{new}} > 0$, or $|s_{\text{old}} - s_{\text{new}}| \geq \tau s_{\text{new}}$, set $s_{\text{old}} = s_{\text{new}}$, and repeat Steps 2-4.
5. Let $\hat{\mathbf{Y}}$ be a $H \times n$ matrix consisting of the elements $\hat{y}_{hj} = y_{hj} - e_{hj}$.
6. Estimate the parameters of the model $y_{hj} = \mu + \alpha_h + \beta_j + \varepsilon_{hj}$ by
   (a) computing the medians r_1 and c_1 of the values in the first row and the first column of $\hat{\mathbf{Y}}$, respectively,
   (b) and setting

$$\hat{\alpha}_h = \hat{y}_{h1} - c_1, \quad \hat{\beta}_j = \hat{y}_{1j} - r_1, \quad \text{and} \quad \hat{\mu} = \hat{y}_{11} - \hat{\alpha}_1 - \hat{\beta}_1.$$

The expression value x_{ij} of probe set i and sample j is then given by

$$x_{ij} = \hat{\mu}^{(i)} + \hat{\beta}_j^{(i)}.$$

Note that the RMA signals are already $\log_2$-scaled, whereas the outcomes of MAS 5.0 are expression values on original scale.

3.4 Modifications of RMA

Drawbacks of median polish are that this procedure, on the one hand, does not provide estimates for the standard errors, and on the other hand, is only applicable to a probe set i, $i = 1, \ldots, m$, if none of the probe intensities $PM_{hj}^{(i)}, h = 1, \ldots, H_i, j = 1, \ldots, n$, is missing. In Section 3.4.1, a summarization method is described that does not have these drawbacks.

In Section 3.4.2, a background correction method alternatively to the convolution model used in RMA is presented that employs the MMs to estimate the non-specific binding affecting the PMs.

3.4.1 Probe Level Model

Instead of median polish, robust regression using M-estimators (Huber, 1981) can be employed to fit the probe level model (PLM)

$$\log_2 PM_{hj}^{(i)} = \alpha_h^{(i)} + \beta_j^{(i)} + \varepsilon_{hj}^{(i)}, \tag{3.3}$$

for each probe set i, $i = 1, \ldots, m$, where

- the probe effects $\alpha_h^{(i)}$, $h = 1, \ldots, H_i$, are constrained by $\sum_h^{H_i} \alpha_h^{(i)} = 0$,
- the expression values x_{ij} are given by the chip effects $\beta_j^{(i)}$, $j = 1, \ldots, n$,
- the errors $\varepsilon_{hj}^{(i)}$ are assumed to be independently and identically distributed.

Fitting such a model can be considered as a weighted linear regression in which the observations are iteratively reweighted based on their standardized residuals (see Algorithm 3.5). The larger the absolute values of the standardized residuals, the smaller are the weights such that outliers have (almost) none influence on the estimation of the parameters $\boldsymbol{\beta} = \begin{bmatrix} \alpha_1 & \ldots & \alpha_{H_i} & \beta_1 & \ldots & \beta_m \end{bmatrix}'$. (For a short introduction to M-estimation and its connection to weighted linear regression, see Appendix D.2.)

Algorithm 3.5 (Robust Linear Regression)

Given a response vector $\mathbf{y}$ of q observations, a $q \times p$ design matrix $\mathbf{Z}$, a weighting function w, a maximum number K_{iter} of iterations, and the tolerance τ for convergence, the parameter vector $\boldsymbol{\beta}$ of the linear model $\mathbf{y} = \mathbf{Z}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ is robustly estimated as follows.

1. Initially, estimate $\boldsymbol{\beta}$ by $\hat{\boldsymbol{\beta}}^{(0)} = (\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{y}$, and set $\hat{\boldsymbol{\varepsilon}}^{(0)} = \mathbf{y} - \mathbf{Z}\hat{\boldsymbol{\beta}}^{(0)}$.

2. For $k = 1, 2, \ldots,$

   (a) compute the standardized residuals $\hat{\mathbf{u}}^{(k-1)} = \hat{\boldsymbol{\varepsilon}}^{(k-1)} / \hat{s}^{(k-1)}$ with scaling parameter $\hat{s}^{(k-1)} = 1.4826 \cdot \text{median}\left|\hat{\boldsymbol{\varepsilon}}^{(k-1)}\right|$,

   (b) construct the $q \times q$ diagonal weight matrix $\mathbf{W}^{(k-1)}$ with diagonal elements $w_{\ell\ell}^{(k-1)} = w\left(\hat{u}_\ell^{(k-1)}\right)$,

   (c) update the estimate for $\boldsymbol{\beta}$ by $\hat{\boldsymbol{\beta}}^{(k)} = \left(\mathbf{Z}'\mathbf{W}^{(k-1)}\mathbf{Z}\right)^{-1}\mathbf{Z}'\mathbf{W}^{(k-1)}\mathbf{y}$,

   (d) set $\hat{\boldsymbol{\varepsilon}}^{(k)} = \mathbf{y} - \mathbf{Z}\hat{\boldsymbol{\beta}}^{(k)}$,

   (e) stop if $k = K_{\text{iter}}$, or if

$$\frac{\left(\hat{\boldsymbol{\varepsilon}}^{(k-1)} - \hat{\boldsymbol{\varepsilon}}^{(k)}\right)'\left(\hat{\boldsymbol{\varepsilon}}^{(k-1)} - \hat{\boldsymbol{\varepsilon}}^{(k)}\right)}{\max\left\{10^{-16}, \left(\hat{\boldsymbol{\varepsilon}}^{(k-1)}\right)'\hat{\boldsymbol{\varepsilon}}^{(k-1)}\right\}} \leq \tau.$$

The results of a comparison of Huber's weighting function

$$w_{\text{H}}(u) = I\big(|u| \leq 1.345\big) + \frac{1.345}{|u|} \cdot I\big(|u| > 1.345\big) \tag{3.4}$$

with the Geman-McClure function $w_{\text{GM}}(u) = (1 + u^2)^{-2}$ and Tukey's biweight function $w_{\text{TB}}(u) = \big(1 - (u/4.6851)^2\big)^2 \cdot I\left(|u| \leq 4.6851\right)$ carried out by Bolstad (2004) indicate that all three approaches fit the models almost equally well. To be in concordance with the standard `R` function `rlm` for fitting robust linear models, Bolstad (2004) thus uses (3.4) as default in the `R` function `fitPLM` for fitting the probe level models (3.3).

3.4.2 Base Composition Based Background Correction

Each base at any position in a PM sequence affects the intensity of this probe (Wu et al., 2004). On the one hand, the higher the GC-content, i.e. the number of the bases G and C in the sequence, the stronger is the hybridization, since G and C are connected via three hydrogen bonds, while A and T are joined by two hydrogen bonds (see Figure 2.1 on page 10). On the other hand, only the pyrimidines C and U are labeled with fluorescent dye which might either impede hybridization if too many bases are labeled, or prevent the shining of sequences that strongly bind if too few bases are labeled (Naef and Magnasco, 2003).

To investigate the effect of the base composition on the probe intensities, Naef and Magnasco (2003) model the PM intensities by a sum over the position-dependent base effects $\theta_{k\ell}$, $k \in \{A, T, C, G\}$, $\ell = 1, \ldots, 25$. The resulting least square estimates $\hat{\theta}_{k\ell}$ are shown in Figure 3.6.

Based on this idea and the results of a few other microarray experiments, Wu et al. (2004) propose an alternative background correction step for RMA. In this modified approach called GCRMA, they assume for any probe pair that

$$PM_{hj}^{(i)} = O_j^{\mathrm{PM}} + N_{hij}^{\mathrm{PM}} + S_{hij} \quad \text{and} \quad MM_{hj}^{(i)} = O_j^{\mathrm{MM}} + N_{hij}^{\mathrm{MM}} + \varphi_{hij} S_{hij},$$

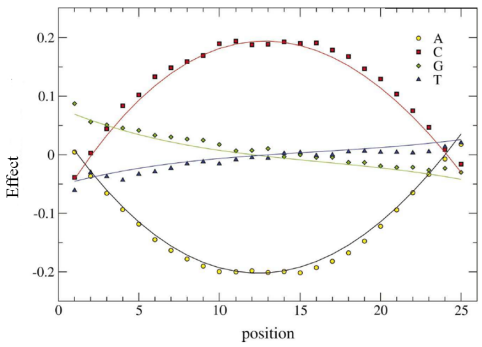

FIGURE 3.6. Position-dependent effects of the four bases A, T, C and G on the probe intensities. (*Source*: Naef and Magnasco, 2003, Figure 3)

where the optical noise O_j, $j = 1, \ldots, n$, follows a lognormal distribution,

$$\begin{bmatrix} \ln N_{hij}^{\mathrm{PM}} \\ \ln N_{hij}^{\mathrm{MM}} \end{bmatrix} \sim \mathrm{N}\left(\begin{bmatrix} \nu_{hij}^{\mathrm{PM}} \\ \nu_{hij}^{\mathrm{MM}} \end{bmatrix}, \sigma_j^2 \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix} \right) \tag{3.5}$$

is the noise based on non-specific binding, S_{hij} is the signal of interest, and $0 < \varphi_{hij} < 1$ takes into account that MMs can also measure specific binding.

For each chip j, $j = 1, \ldots, n$, Wu et al. (2004) estimate the parameters by ad-hoc approaches: φ is set to zero, since following Wu et al. (2004) this has only a small impact on the results of the approach. The optical noise is assumed to be constant (since the variance of O_j is almost zero), and calculated by $\hat{O}_j = \min_{i,h} \left\{ PM_{hj}^{(i)}, MM_{hj}^{(i)} \right\} - 1$. The parameters in (3.5) are estimated by firstly fitting a loess curve f_j (Cleveland and Devlin, 1988) through the scatter plot of $\ln\left(MM_{hj}^{(i)} - \hat{O}_j\right)$ vs. $\hat{\lambda}_{hi}^{\mathrm{MM}}$, where the probe affinities $\hat{\lambda}$ are determined by summing over the base effects $\hat{\theta}_{k\ell}$ corresponding to the respective probe sequence. (These base effects slightly differ from the $\hat{\theta}_{k\ell}$ shown in Figure 3.6, since the model Wu et al., 2004, use differs slightly from the model of Naef and Magnasco, 2003). Secondly, σ_j is set to the MAD of the negative residuals resulting from this regression, and the means are estimated by $\hat{\nu}_{hij}^{\mathrm{PM}} = f_j\left(\hat{\lambda}_{hi}^{\mathrm{PM}}\right)$ and $\hat{\nu}_{hij}^{\mathrm{MM}} = f_j\left(\hat{\lambda}_{hi}^{\mathrm{MM}}\right)$.

The background corrected intensity of each PM is then computed by a truncated maximum likelihood estimator for S_{hij} given by

$$\hat{S}_{hij} = \begin{cases} PM_{hj}^{(i)} - \hat{O}_j - \hat{N}_{hij}^{\mathrm{PM}}, & \text{if } PM_{hj}^{(i)} - \hat{O}_j - \hat{N}_{hij}^{\mathrm{PM}} > \tau \\ \tau, & \text{if } PM_{hj}^{(i)} - \hat{O}_j - \hat{N}_{hij}^{\mathrm{PM}} \leq \tau \end{cases}, \tag{3.6}$$

where $\hat{N}_{hij}^{\mathrm{PM}} = \exp\left(0.7 \cdot \ln\left(MM_{hj}^{(i)} - \hat{O}_j\right) + \nu_{hij}^{\mathrm{PM}} - 0.7 \cdot \nu_{hij}^{\mathrm{MM}} - (1 - 0.7^2)\,\sigma_j^2\right)$, and τ is the minimum value allowed for S_{hij}. In the `R` function `gcrma`, this value is set by default to $\tau = 6$.

In this implementation of GCRMA, two further corrections are made that are not mentioned in Wu et al. (2004). First, the linear model

$$\log_2 PM_{hj}^{(i)} = \gamma_0 + \gamma_1 \hat{\lambda}_{hi}^{\mathrm{PM}} + \epsilon_{hj}^{(i)}$$

is fitted for a randomly chosen subset of the PM intensities. For each chip j, $j = 1, \ldots, n$, the PM intensities background corrected by (3.6) are then additionally adjusted by setting $\log_2 PM_{hj}^{(i)}$ to

$$s_{hij} = \log_2 PM_{hj}^{(i)} - \hat{\gamma}_1 \hat{\lambda}_{hi}^{\mathrm{PM}} + n_{\mathrm{PM}}^{-1} \sum_{k=1}^{m} \sum_{\ell=1}^{H_i} \hat{\gamma}_1 \hat{\lambda}_{k\ell}^{\mathrm{PM}}.$$

Afterwards, the final background corrected PM values are determined by setting $PM_{hj}^{(i)}$ to

$$S_{hij}^{\mathrm{new}} = \exp\left\{ n_{\mathrm{PM}}^{-1} \sum_{k,\ell} \ln PM_{kj}^{(\ell)} + 1.15 \left(\ln PM_{hj}^{(i)} - n_{\mathrm{PM}}^{-1} \sum_{k,\ell} \ln PM_{kj}^{(\ell)} \right) \right\}.$$

3.5 Probe Logarithmic Intensity Error Estimation

In July 2004, MAS 5.0 was replaced by PLIER (**P**robe **L**ogarithmic **I**ntensity **ER**ror estimation) as standard Affymetrix preprocessing algorithm (Affymetrix, 2005). Even though Affymetrix (2005) recommends to normalize either the probe intensities or the expression values, PLIER just consists of a summarization step in which the multi-chip model

$$\varepsilon_{hij}^{\mathrm{PM}} PM_{hj}^{(i)} - \varepsilon_{hij}^{\mathrm{MM}} MM_{hj}^{(i)} = \alpha_h^{(i)} \beta_j^{(i)} \tag{3.7}$$

is fitted for each probe set i, $i = 1, \ldots, m$, where $\alpha_h^{(i)} \geq 0$ and $\beta_j^{(i)} \geq 0$ are the probe and chip effects, respectively, and $\varepsilon_{hij}^{\mathrm{PM}} > 0$ and $\varepsilon_{hij}^{\mathrm{MM}} > 0$ are the multiplicative errors of the PMs and MMs, respectively.

To simplify computation, (3.7) is fitted under the constraint that

$$\ln\left(\varepsilon_{hij}^{\mathrm{PM}}\right) = -\ln\left(\varepsilon_{hij}^{\mathrm{MM}}\right) \tag{3.8}$$

leading to

$$\varepsilon_{hij}^{\mathrm{PM}} = \frac{\alpha_h^{(i)} \beta_j^{(i)} + \sqrt{\left(\alpha_h^{(i)} \beta_j^{(i)}\right)^2 + 4 PM_{hj}^{(i)} MM_{hj}^{(i)}}}{2 PM_{hj}^{(i)}}. \tag{3.9}$$

Using (3.9), estimates for the probe and the chip effects of probe set i are computed by

$$\arg\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \sum_{j=1}^{n}\sum_{h=1}^{H_i} \rho_{\mathrm{GM}}\Big(\ln\left(\varepsilon_{hij}^{\mathrm{PM}}\right),\eta\Big), \tag{3.10}$$

where $\rho_{\mathrm{GM}}(z,\eta) = z^2/(1+z^2/\eta)$ is the ρ function of Geman-McClure (see Appendix D.2) with η being a tuning constant that bounds $\rho_{\mathrm{GM}}(z,\eta)$ to be smaller than η for any z. In the `R` wrapper `justPlier` for the PLIER code of Affymetrix, η is by default set to 0.15.

As in Section 3.4.1, the expression value x_{ij} of probe set i and sample j is then given by $\hat{\beta}_j^{(i)}$. However, contrary to Section 3.4.1, x_{ij} is not on $\log_2$-scale.

3.6 PLIER Like Algorithms

Since the assumption (3.8) can lead to suboptimal solutions of (3.10), Wei-Min Liu, Roche Molecular Systems, Alameda, CA, USA, decided to implement his own version of PLIER called PLA (**P**lier **L**ike **A**lgorithm; Liu, 2004) throughout this thesis. Instead of using this constraint, Liu (2004) assumes that the multiplicative errors $\varepsilon_{hij}^{\mathrm{PM}}$ and $\varepsilon_{hij}^{\mathrm{MM}}$ are typically almost 1, and that thus

$$\ln\left(\varepsilon_{hij}^{\mathrm{PM}}\right) \approx \varepsilon_{hij}^{\mathrm{PM}} - 1 \quad \text{and} \quad \ln\left(\varepsilon_{hij}^{\mathrm{MM}}\right) \approx \varepsilon_{hij}^{\mathrm{MM}} - 1. \tag{3.11}$$

Using (3.7) and (3.11), Liu (2004) derives the two equations

$$\ln\left(\varepsilon_{hij}^{\mathrm{PM}}\right) = \frac{M_{hj}^{(i)} - 1 + \alpha_h^{(i)}\beta_j^{(i)}/PM_{hj}^{(i)}}{1+\left(M_{hj}^{(i)}\right)^2} \quad \text{and} \quad \ln\left(\varepsilon_{hij}^{\mathrm{MM}}\right) = -M_{hj}^{(i)}\ln\left(e_{hij}^{\mathrm{PM}}\right)$$

with $M_{hj}^{(i)} = MM_{hj}^{(i)}/PM_{hj}^{(i)}$, and computes the estimates for the probe and the chip effects by

$$\arg\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \sum_{j=1}^{n}\sum_{h=1}^{H_i}\left(\rho_{\mathrm{GM}}\Big(\ln\left(\varepsilon_{hij}^{\mathrm{PM}}\right),\eta\Big) + \rho_{\mathrm{GM}}\Big(\ln\left(\varepsilon_{hij}^{\mathrm{MM}}\right),\eta\Big)\right) \tag{3.12}$$

under the constraint that $\sum_{h=1}^{H_i}\alpha_h^{(i)} = H_i$. The unnormalized expression values

for probe set i, $i = 1, \ldots, m$, are then given by

$$x_{ij} = \begin{cases} \log_2 \hat{\beta}_j^{(i)}, & \text{if } \hat{\beta}_j^{(i)} \geq 1 \\ \left(\hat{\beta}_j^{(i)} - 1\right) / \log(2), & \text{if } 0 \leq \hat{\beta}_j^{(i)} < 1 \end{cases} \tag{3.13}$$

which are afterwards quantile normalized (see Algorithm 3.3) to obtain the PLA signal.

In a second approach presented in Algorithm 3.6, Liu (2004) additionally adjusts for high probe effects.

Algorithm 3.6 (PLIER Like Algorithm+)

Let $\hat{\beta}_j^{(i)}$, $i = 1, \ldots, m$, $j = 1, \ldots, n$ be the solutions of (3.12).

1. For each probe set i and chip j, compute the median v_{ij} of $PM_{hj}^{(i)}$, $h = 1, \ldots, H_i$.

2. For each chip j, determine the 95% quantile $q_{0.95}^{(j)}$ and the 98% quantile $q_{0.98}^{(j)}$ of the medians v_{ij}, $i = 1, \ldots, m$.

3. Set $\hat{\beta}_j^{(i)}$ to

$$\hat{\beta}_j^{(i)\text{new}} = \begin{cases} \hat{\beta}_j^{(i)}, & \text{if } v_{ij} \leq q_{0.95}^{(j)} \\ v_{ij}, & \text{if } v_{ij} \geq q_{0.98}^{(j)} \\ \hat{\beta}_j^{(i)} + \omega_{ij}\left(v_{ij} - \hat{\beta}_j^{(i)}\right) & \text{otherwise} \end{cases},$$

   where $\omega_{ij} = \left(v_{ij} - q_{0.95}^{(j)}\right) / \left(q_{0.98}^{(j)} - q_{0.95}^{(j)}\right)$ if $q_{0.98}^{(j)} \neq q_{0.95}^{(j)}$, and otherwise $\omega_{ij} = 0.5$,

4. determine the unnormalized expression values by (3.13), and quantile normalize them to generate the PLA+ signals.

3.7 Preprocessing of SNP Microarrays

On Affymetrix genotype arrays, each allele of a SNP is represented by ten probe pairs (see Section 2.2.2). Therefore, Rabbee and Speed (2006) propose a procedure called RLMM (**R**obust **L**inear **M**odel with **M**ahalanobis distance) in which two allele-specific RMA signals β_A and β_B (see Section 3.3) are computed for each SNP, where the background correction step of RMA is omitted. To genotype a SNP with RMA signal vector $\boldsymbol{\beta} = \begin{bmatrix}\beta_A & \beta_B\end{bmatrix}'$, a training set is used to determine the mean vector $\boldsymbol{\mu}_c$ of the two RMA signals and the corresponding covariance matrix $\mathbf{S}_c$ for each genotype c. Afterwards, the squared Mahalanobis distances

$$d_c^2 = (\boldsymbol{\beta} - \boldsymbol{\mu}_c)' \, \boldsymbol{S}_c^{-1} \, (\boldsymbol{\beta} - \boldsymbol{\mu}_c)$$

between $\boldsymbol{\beta}$ and the center $\boldsymbol{\mu}_c$ of each of the three genotype groups is computed, and the genotype of the SNP is specified by $\arg\min_c \left\{ d_c^2 \right\}$.

The latest Affymetrix genotype calling algorithm referred to as BRLMM (**B**ayesian **RLMM**; Affymetrix, 2006) is a modification of RLMM in which an ad-hoc Bayesian approach is employed to derive posterior estimates of the group means and covariances.

Another refinement of RLMM is proposed by Carvalho et al. (2006). In this procedure called CRLMM (**C**orrected **R**obust **L**inear **M**odel with **M**aximum likelihood based distance classifier), the PM intensities are corrected for sequence effects, and the SNPs are afterwards genotyped using maximum likelihood based approaches.

Chapter 4

Comparison of Preprocessing Methods

4.1 Introduction

Since preprocessing is an important step in the analysis of Affymetrix microarray data, originally Cope et al. (2004), and in its latest version Irizarry et al. (2006) provide a webtool based on the `R` package `affycomp` that enables the comparison of preprocessing procedures in their application to the probe data from two microarray experiments. Some of the probe sets composing these two data sets available at http://affycomp.biostat.jhsph.edu consist of probes that have been spiked-in at known concentrations such that it is, e.g., known which probe sets are differentially expressed. After having applied the preprocessing method of interest to these data sets, the resulting expression values can be uploaded to this web page at which several statistics for assessing the quality of the signals are computed automatically and presented in a table currently containing the results of the applications of more than 30 preprocessing methods – many of them in several versions.

Employing the same data sets as Cope et al. (2004), Bolstad (2004) compares not only a subset of these preprocessing procedures as a whole, but also the

different steps of these methods separately.

The goal of a study of Roche Diagnostics, Penzberg, is the identification of the preprocessing method among a set of procedures consisting of the in-house algorithms PLA and PLA+, the Affymetrix approach MAS 5.0, and the academic alternatives RMA and PLM that leads to the best data reduction. This approach will therefore be used as the standard preprocessing procedure in upcoming projects of Roche Diagnostics concerned with Affymetrix microarrays. For this study, a data set containing the probe intensities of 38 Affymetrix HG-U133_Plus_2 chips is available, where 18 of these samples come from a CRCA (colorectal cancer) project, and 20 from a BRCA (breast cancer) project (for more details on the data, see Appendix A.1). Each of the five preprocessing methods is applied in its default setting to this data set, and the procedure are compared using the prespecified criteria visual comparison, linearity and correlation of control probe sets, signal-to-noise ratio, differential expression, and multivariate analysis. These five criteria also build the structure of this chapter.

In Schwender and Belousov (2006), not PLA and PLA+, but PLIER and a version of PLIER called qPLIER throughout this thesis in which PLIER is applied to the quantile normalized probe intensities are compared with the other preprocessing methods. In this thesis, we therefore also examine the performance of the latter two algorithms. Since the background correction method proposed by Wu et al. (2004) should improve the estimation of the RMA signals, we also consider versions of RMA and PLM in this thesis in which this approach is employed in the background correction step. To be in concordance with the study of Roche Diagnostics, these four additional preprocessing procedures are examined only in their respective default setting. All preprocessing methods used in the comparisons presented in the following sections are summarized in Table 4.1.

For the RMA approaches, the expression values of the 38 samples are generated using the `R` function `just.rmaplm` (see Appendix C.3). The PLIER and

TABLE 4.1. Summary of the preprocessing procedures compared in Chapter 4.

	Background Correction	Normalization	Summarization Model	Summarization Procedure	Expression Value Estimate	Expression Value Scale
RMA	Convolution Model $PM = N + S$	Quantile Normalize Probe Intensities	Multi-Chip Model $\log_2 PM = (\mu+)\,\alpha + \beta + \varepsilon$	Median Polish	$\hat{\mu} + \hat{\beta}$	$\log_2$
PLM				Robust Linear Model	$\hat{\beta}$	
GCRMA	Based on Base Composition			Median Polish	$\hat{\mu} + \hat{\beta}$	
GCPLM				Robust Linear Model	$\hat{\beta}$	
MAS 5.0	Spatial	Scale Expression Values	Single Chip, Robust Mean of $PM - IM$	Tukey's One-Step Biweight Estimate	$T_{\mathrm{B}}(PM - IM)$	Original
PLA	None	Quantile Normalize Expression Values	Multi-Chip Model $\varepsilon^{\mathrm{PM}} PM + \varepsilon^{\mathrm{MM}} MM = \alpha\beta$	Minimize under assumptions $\ln(\varepsilon^{\mathrm{PM}}) \approx \varepsilon^{\mathrm{PM}} - 1$, $\ln(\varepsilon^{\mathrm{MM}}) \approx \varepsilon^{\mathrm{MM}} - 1$	Truncated $\hat{\beta}$	$\log_2$
PLA+					PLA signal, at high end: Median of PMs	
PLIER		None		Minimize with constraint $\ln(\varepsilon^{\mathrm{PM}}) = -\ln(\varepsilon^{\mathrm{MM}})$	$\hat{\beta}$	Original
qPLIER		Quantile Normalize Probe Intensities				

qPLIER signals are computed by employing `justPLIER` (see Appendix C.3), whereas MAS 5.0, PLA and PLA+ are applied to the 38 array using the internal Roche software GX.

While the signals of the RMA and the PLIER-like approaches are already on $\log_2$-scale, the expression values of MAS 5.0, PLIER and qPLIER are additionally $\log_2$-transformed. (For PLIER and qPLIER, this is done by `justPLIER`.)

4.2 Visual Comparison

As a first visual comparison, the Euclidean distance between the expression values for any pair of preprocessing methods is computed, and a hierarchical cluster analysis using complete linkage is performed on these dissimilarities. The resulting dendrogram displayed in Figure 4.1 reveals that there are four pairs of procedures that are each very close to each other: PLIER and qPLIER, PLA and PLA+, RMA and PLM, and GCRMA and GCPLM. The latter two pairs indicate that – at least when using the RMA approaches – the background correction step has a higher impact on the expression values than the summarization step. This has also been noted by Irizarry et al. (2006).

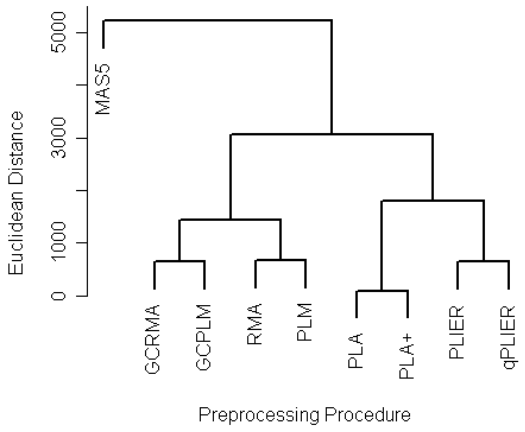

FIGURE 4.1. Dendrogram of the distances between different preprocessing methods generated by hierarchical clustering using the Euclidean distance and complete linkage.

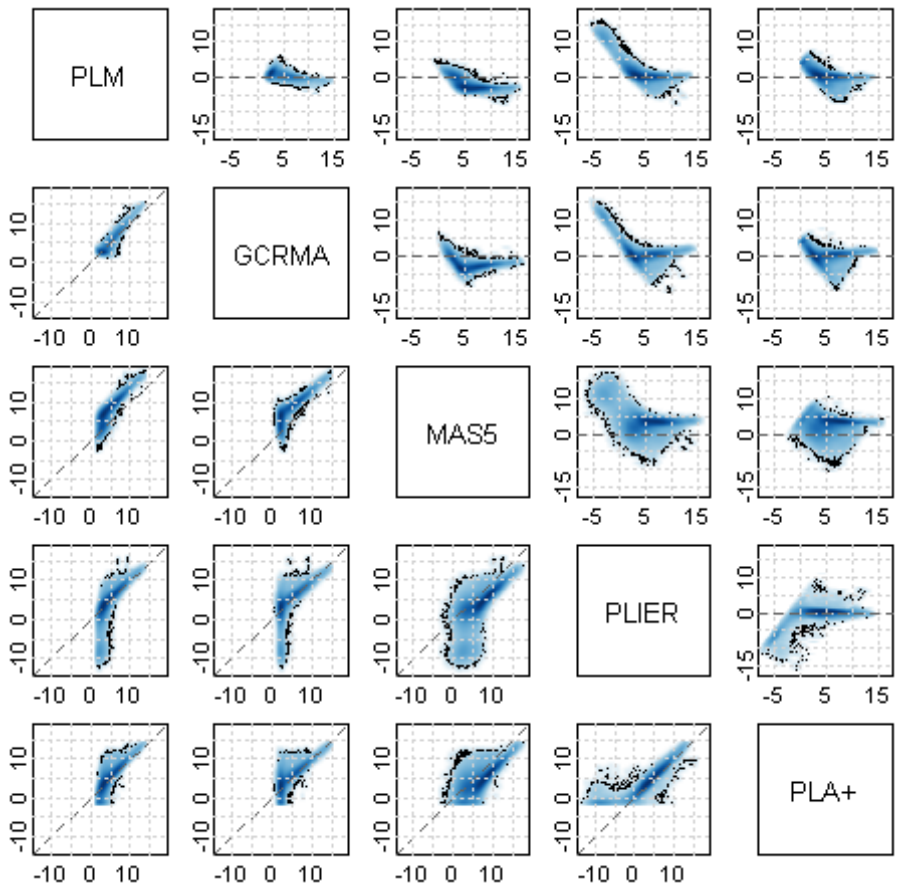

FIGURE 4.2. Pairwise smoothed scatter (lower triangle) and MA (upper triangle) plots of the expression values of five of the preprocessing methods. The darker the color, the higher is the density at this point. The 500 signals with the lowest density are marked by black dots.

Moreover, this dendrogram shows that the three groups of algorithms, i.e. the RMA approaches, the PLIER and PLIER-like methods, and MAS 5.0, are clearly separated from each other. In particular, MAS 5.0 shows a very large distance to the other methods.

In Figure 4.2, a reason for this separation is revealed by the smoothed scatter and MA plots, i.e. plots of $\mathbf{y}$ vs. $\mathbf{x}$, and $\mathbf{m} = \mathbf{y} - \mathbf{x}$ vs. $\mathbf{a} = 0.5\,(\mathbf{y} + \mathbf{x})$ in which not the data points themselves, but the two-dimensional density at each point is shown (Wand and Jones, 1995). Most of the MAS 5.0 signals are larger than the corresponding expression values resulting from any of the other preprocessing

methods which is due to the high target signal used in the normalization step of MAS 5.0. In this step, the 2% trimmed mean of the (not $\log_2$-transformed) expression values of each chip is set to a target signal of 500 (cf. Section 3.2) which is almost tenfold larger than the 2% trimmed mean of the expression values of any of the other methods ranging from 51.12 (PLM) to 67.05 (PLA+), and more than tenfold larger than 45.52, the 2% trimmed mean of the unnormalized MAS 5.0 signals.

For a better visual representation, only the expression values of one of the procedures from each of the above-mentioned pairs is shown in Figure 4.2. (Smoothed scatter and MA plots of the four RMA approaches and the four PLIER methods are displayed in Figure B.1 and B.2, respectively.) This figure additionally reveals that none of the $\log_2$-scaled signals of any of RMA procedure is smaller than zero, whereas about 5% of the expression values of each of the PLIER approaches are negative. These plots also show the effect of the truncation (3.13) in the low intensity region that is applied to the PLA and PLA+ signals, but not used in PLIER and qPLIER.

4.3 Linearity and Correlation of Controls and Housekeeping Genes

Not all probe sets represent a gene. Some of them are employed for quality control. Examples for such controls are, on the one hand, BioB, BioC, BioD, and cre that are used to monitor the hybridization process, and on the other hand, the poly-A probe sets lys, phe, thr, and dap utilized to control the labeling and the hybridization (cf. Section 2.2.1). For each of these controls originated from non-human DNA – BioB, BioC, and BioD come from an E. coli bacterium, and the poly-A probes from a B. subtilis bacterium – at least two probe sets exist: One composed of probes from near the 5' end, and the other containing probes from near the 3' end. (For some of the controls, additionally a set of

probes from the middle region of the DNA sequence is available.)

Since these controls are spiked in at a known concentration (for details, see Affymetrix, 2003), their actual concentration can be compared with their signals estimated by the preprocessing methods to identify the procedure showing the best linear relationship between these values. Thus, the mean of the original scaled, i.e. not $\log_2$-transformed, expression values is method-wise computed for each of the control probe sets. Two linear regressions – one with BioB, BioC, BioD, and cre, and one with the poly-A controls – of the means vs. the concentrations are conducted for each of the preprocessing methods, and the resulting R^2-statistics are employed as a measure for the linearity of the relationship (see Table 4.2).

Another control typically used in the standard Affymetrix quality check is GAPDH. For both GAPDH and BioB, the signals of the set of probes from near the 5' end should be correlated with the expression values of the set of probes from near the 3' end. Therefore, the R^2-statistic of the 5' vs. the 3' probe set

TABLE 4.2. Method-wise computed R^2-statistics as a measure for the linearity and the correlation of the signals of the control probe sets and housekeeping genes.

	Linearity		Correlation		
	BioB, C, D, cre	Poly-A	Housekeepers	BioB	GAPDH
RMA	0.9940	0.9341	0.9375	0.9515	0.7474
PLM	0.9981	0.9346	0.9311	0.9547	0.7532
GCRMA	0.9869	0.9171	0.9379	0.9518	0.7416
GCPLM	0.9878	0.9463	0.9321	0.9525	0.7507
MAS 5.0	0.9861	0.9543	0.9330	0.8475	0.7772
PLIER	0.9923	0.9478	0.9236	0.7912	0.8673
qPLIER	0.9916	0.9483	0.9236	0.9090	0.7759
PLA	0.9908	0.9481	0.9153	0.8513	0.6743
PLA+	0.9974	0.9350	0.9175	0.7429	0.7147

is method-wise computed for both controls as measure of their correlation (see Table 4.2).

GAPDH is one of the 100 human genes on the HG-U133_Plus_2 chip specified by Affymetrix as a housekeeping gene, i.e. as a gene involved in the maintenance of cells. Such genes are always expressed and assumed to be not affected by experimental conditions. If thus the mean of the housekeeping genes is calculated project-wise, then the averages from the CRCA project should be correlated with the means from the BRCA project. Again, the R^2-statistic is determined for each of the preprocessing procedures to measure this correlation.

In Table 4.2, the R^2-statistics of all comparisons are summarized. This table shows that the preprocessing methods perform almost equally well in terms of the linearity of both the poly-A controls and BioB, BioC, BioD, and cre. Only the R^2-statistic of the poly-A signals estimated by GCRMA is a little smaller than the R^2-statistics of the other methods, but with 0.917 still relatively high.

Large differences between preprocessing methods can only be observed in the correlation of the 3' and 5' probe sets of BioB. For this control, the R^2-statistics of the RMA approaches are substantially higher than the R^2-statistics of the other procedure. While PLIER exhibits the second lowest correlation of the BioB probe sets, it outperforms the other methods by far when considering the GAPDH probe sets. In all three correlation comparisons, PLA and PLA+ show the worst performance.

4.4 Signal-to-Noise Ratio

The Signal-to-Noise (S/N) ratio

$$SNR(\mathbf{z}) = \frac{\text{median}(\mathbf{z})}{0.741 \cdot \text{IQR}(\mathbf{z})}$$

of a vector $\mathbf{z}$ of observations can serve as an overall measure of the quality of a signal affected by noise, where $0.741 \cdot \text{IQR}$ is a consistent estimate of the standard deviation at the normal distribution.

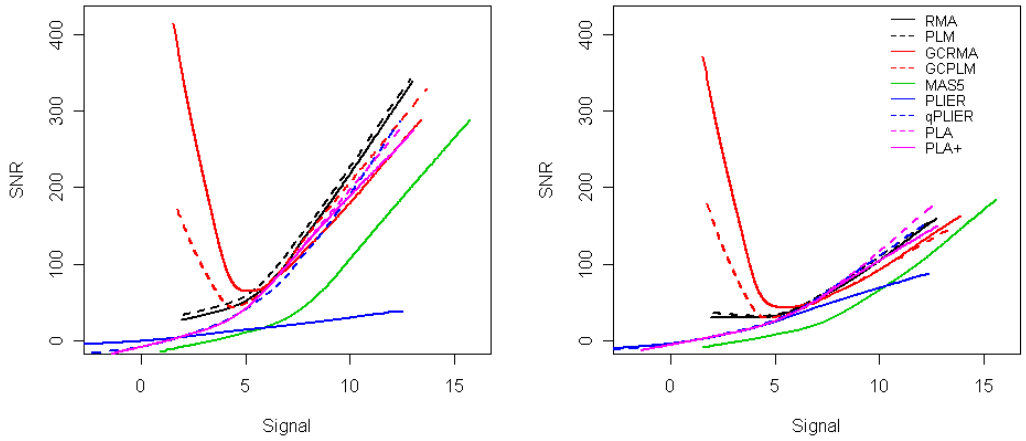

FIGURE 4.3. Loess curves method-wise fitted through the Signal-to-Noise Ratios vs. the median signals of 2,982 potentially interesting cancer-related probe sets. Left Panel: Arrays from the CRCA project. Right Panel: Chips from the BRCA project.

To get an impression of which preprocessing method leads to the least perturbed expression values, the S/N ratio is computed project-wise for each of 2,982 potentially interesting cancer-related probe sets specified prior to this comparison. (This list is based on van't Veer et al., 2002, Eschrich et al., 2005, Wang et al., 2004, and the genes from the Affymetrix Human Cancer G110 array, and has been provided by Anton Belousov, Roche Diagnostics, Penzberg.) These S/N ratios are plotted against the corresponding median signals, and a loess curve (Cleveland and Devlin, 1988) is fitted through the data points. For each of the nine preprocessing procedures, this fitted curve is displayed in Figure 4.3.

This figure reveals that – in particular, in the CRCA project – model-based multi-chip approaches achieve a better precision than the single-chip approach MAS 5.0 when either the probe intensities or the expression values are normalized. PLIER, the only method in which no normalization is done, shows the by far worst precision. The RMA approaches have about the same linear region, i.e. the same region in which the noise is constant, whereas this region is extended (towards the origin) a little by PLA and PLA+.

In the low intensity region, the curves for GCRMA and GCPLM show an abnormal behavior, as the S/N ratio is expected to decrease when the signal

decreases. The reason for this behavior of the S/N ratio of GCRMA and GCPLM is the truncation in (3.6) in which all background corrected PM intensities smaller than $\tau = 6$ (on original scale) are set to τ. The intensities of 23.8% of the PMs from the cancer-related probe sets are affected by this truncation leading to very small IQRs in the low intensity region. In fact, the only reason why none of the probe sets shows a zero IQR is that further background adjustments are made in the `R` function `gcrma` that are not mentioned by Wu et al. (2004) (cf. Section 3.4.2) such that the intensities of the truncated PMs range from 1.46 to 17.90.

If probe sets exhibiting more than 20% (GCRMA) or 30% (GCPLM) truncated background corrected PMs, respectively, are excluded from the computation of the S/N ratio, the fitted curves for GCRMA and GCPLM will behave as expected (cf. Figure B.3 and B.4 in Appendix B.1).

4.5 Differential Expression

The goal of preprocessing is to effectively reduce the 22 probe intensities representing a gene to one expression value. This summarization should be performed for each of tens of thousands of probe sets on tens or even hundreds of microarrays. This immense data reduction leads to a loss of information that in turn results in a reduced number of genes with convincingly large test statistic when testing for differential expression (Efron et al., 2001). The number of identified genes, i.e. the number of rejected null hypotheses, can thus serve as a criterion for which preprocessing method leads to the best data reduction. Moreover, the clearer the separation between the considered groups, the easier is the identification of genuinely affected genes.

Since in the two class case typically a t-statistic is calculated, it furthermore would be preferable if the assumptions for the t-test are fulfilled such that a time-consuming estimation of the null distribution can be avoided.

For the comparisons in this section, only the "best" probe sets are used, i.e.

the genes that meet the following two conditions:

- To ensure that the probes in the probe set belong to the gene of interest, the probe set must represent at least one verified RefSeq sequence. (RefSeq, http://www.ncbi.nlm.nih.gov/RefSeq, is the public repository that provides the best non-redundant and comprehensive set of (mRNA) sequences.)

- Since RNA degrades from the 5' to the 3' end, the probe set has to consist of oligos that are less than 600 base pairs away from the 3' end to safeguard against potential degradation problems.

(Note that this list of 13,674 probe sets has been generated in 2004 such that the used information might be outdated, and a more recent list might contain more probe sets.)

For each probe set i, $i = 1, \ldots, 13{,}674$, Welch's t-statistic t_i is computed to test if the mean signal of the BRCA samples differ from the mean expression value of the CRCA samples. The group labels are permuted 10,000 times, and the permuted t-statistics t_{ib}, $b = 1, \ldots, 10{,}000$, are calculated. Afterwards, the already Bonferroni adjusted p-value of probe set i is determined by

$$p_i = \frac{1}{10,000} \sum_{k=1}^{13,674} \sum_{b=1}^{10,000} I\Big(|t_i| \leq |t_{kb}|\Big)$$

(see Appendix D.3.1 for a short discussion of this p-value). In Table 4.3, the numbers of probe sets

- for which $p_i \leq 0.05$,

- identified when using the signals from the respective preprocessing method, but not when employing the MAS 5.0 signals,

- for which the minimum value in one of the groups is larger than the maximum signal in the other group,

- exhibiting a p-value of the Fligner-Killeen test for equal group variances that is smaller than or equal to 0.01,

– for which at least one of the two groups shows a p-value of the Shapiro-Wilk test for normality smaller than or equal to 0.025

are summarized (for a description of the Fligner-Killeen and the Shapiro-Wilk test, see Appendix D.3.2 and D.3.3, respectively).

PLM and PLA identify the most differentially expressed genes closely followed by PLA+, whereas MAS 5.0 and PLIER detect the smallest numbers of probe sets. The base composition based background correction reduces not only the number of identified genes, but also the number of probe sets with perfectly separated groups. While PLA leads to the most genes with perfectly separated groups, MAS 5.0 and PLIER also show the worst performance in this comparison. The only advantage of these two methods seems to be that they exhibit the least heterogeneity problems. Considering non-normality, RMA and PLM lead to a much smaller number of genes showing problems with this assumption of

TABLE 4.3. Numbers of differentially expressed genes, of probe sets identified by the respective preprocessing method but not by MAS 5.0, of genes leading to perfectly separated groups, and of genes with homogeneity or normality problems, respectively, for each of the preprocessing procedures.

	Identified Genes	Not Identified by Mas5.0	Perfectly Separated	Non-Homogeneity	Non-Normality
RMA	4,691	1,200	2,570	1,796	1,640
PLM	5,107	1,548	2,884	2,128	1,415
GCRMA	3,973	639	2,427	2,696	8,185
GCPLM	4,703	1,154	2,750	1,961	3,124
MAS 5.0	3,734	–	2,010	1,494	4,847
PLIER	3,264	651	1,299	757	5,252
qPLIER	4,830	1,477	2,731	2,231	5,913
PLA	5,106	1,585	2,920	2,185	4,337
PLA+	5,073	1,574	2,864	2,150	4,335

the parametric t-test than any of the other methods. The number of probe sets for which the normality assumption is justifiable increases substantially when the base composition based background correction procedure is used instead of the convolution model – in particular, if additionally median polish is employed for summarization.

4.6 Principle Component Analysis

To reduce the high-dimensionality of gene expression data, a principle component analysis (PCA) can be applied to these data (Johnson and Wichern, 1998). The goal of a PCA is to identify linear combinations of genes that best explain the variability in the data set. The smaller the number of such principle components required to capture most of this variability, the more parsimonious is the PCA representation.

For each preprocessing method, PCA is thus applied, on the one hand, to the signals themselves, and on the other hand, to the signals that are project-wise mean-centered such that the mean expression value of probe set i, $i = 1, \ldots, 54{,}675$, is zero for both the BRCA and the CRCA samples.

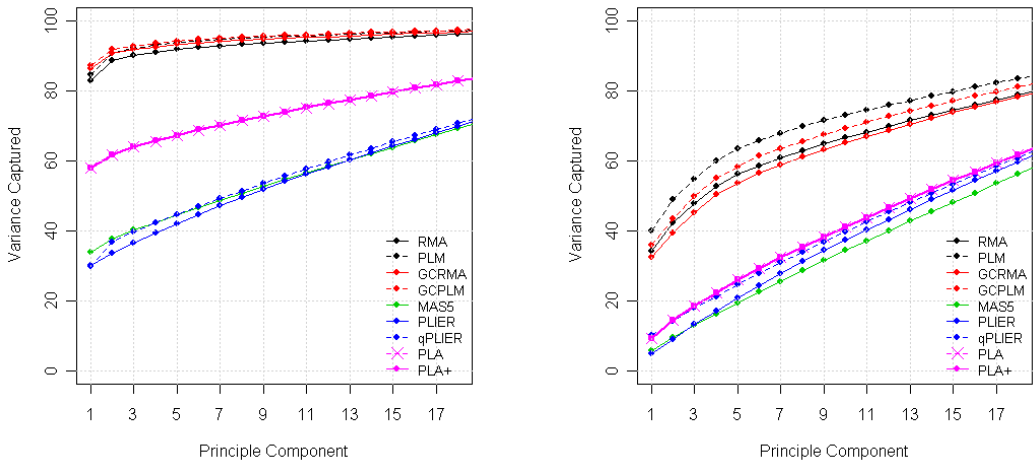

FIGURE 4.4. Scree plots of the first 18 principle components computed for both the signals themselves (left panel) and the project-wise mean-centered expression values (right panel) for each of the preprocessing methods.

The scree plots in Figure 4.4 reveal that in the applications to the signals themselves, MAS 5.0, qPLIER and PLIER need many more principle components to explain the same proportion of variability in the data than PLA and PLA+ which in turn require many more components than the RMA methods. If the project effect is removed the distance between PLA(+) and the Affymetrix approaches will shrink almost to zero, whereas the RMA procedures still show the most parsimonious PCA representation. In this second application, the RMA approaches in which probe level models have been fitted slightly outperform the methods in which median polish has been used for summarization.

4.7 Conclusions

In this chapter, we have compared nine preprocessing procedures to identify the approach that provides the best reduction of the typically eleven pairs of probe intensities per gene and sample to one expression value.

The single-chip method MAS 5.0 and the multi-chip approach PLIER in which the normalization step is avoided show the worst performance in almost any of the comparisons: They lead to the lowest S/N ratios, they allow to identify the smallest number of differentially expressed genes, and they need many more principle components than the other procedures to explain the same proportion of variation. Applying PLIER to quantile normalized probe intensities seems to compensate the former two problems, and makes qPLIER a serious competitor for the other preprocessing methods.

From the two groups of procedures, PLM seems to be the best RMA approach, and PLA the best PLIER and PLIER-like method, since they both outperform – at least slightly – the other approaches.

In a comparison of these two methods, PLM requires less principle components to capture the same proportion of variation and shows a slightly larger S/N ratio, whereas PLA identifies virtually the same number of genes and exhibits a larger number of perfectly separated groups. PLA, however, also shows larger

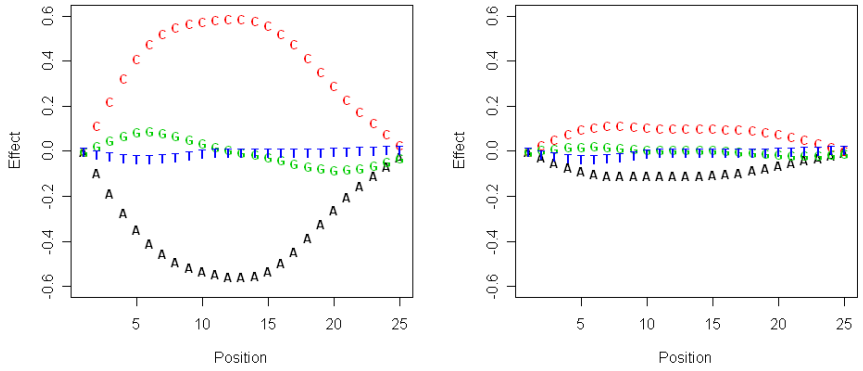

FIGURE 4.5. The base effects used in GCRMA and GCPLM (left panel), and estimated using the 38 Affymetrix HG-U133_Plus_2 chips (right panel).

normality problems and the smallest correlations of both the housekeeping genes and the GAPDH probe sets.

Therefore, the result of the study for Roche Diagnostics (cf. Section 4.1) to choose PLM as standard preprocessing algorithm still holds if the set of considered methods is extended by adding other (popular) approaches.

Since the base component based background method should actually improve RMA, it is a little bit surprising that GCRMA and GCPLM result in worse data reductions than RMA and PLM, respectively. This particularly leads to smaller numbers of genes detected as differentially expressed. A reason for this might be that the position-dependent base effects θ (see Section 3.4.2) have been computed by Wu et al. (2004) once on a set of Affymetrix HG-U133A chips, and are used as standard in any application of this background method – no matter which type of Affymetrix microarray is analyzed.

To check whether the base effects in our data set are similar to the base effects of Wu et al. (2004), these effects are computed as described by Wu et al. (2004). As Figure 4.5 reveals, the base effects estimated by the Affymetrix HG-U133_Plus_2 data show a similar pattern as the base effects used in the background correction, but are much smaller.

Using the base effects displayed in the right panel of Figure 4.5 and an empirical Bayes approach also proposed by Wu et al. (2004) as an alternative

to the maximum likelihood estimator (3.6) might improve the performance of GCRMA and GCPLM.

Finally, we would like to remark that even though the comparisons presented in Bolstad (2004) and in this thesis show that it is advantageous to borrow strength from the whole set of chips by fitting multi-chip models, this only applies to experiments in which more than just a few microarrays are available. If one analyses just three or four chips, a single-chip approach might be better. The same applies to quantile normalization: It works pretty fast, even for a large set of arrays, and seems to provide a good normalization. But it should not be employed in an experiment consisting of three or four chips. In this case, other normalization approaches such as cyclic loess (Bolstad et al., 2003) are to prefer.

Chapter 5

Preprocessing of a Huge Number of Microarrays Using R

5.1 Introduction

The BioConductor project (http://www.bioconductor.org) provides a large number of packages for the analysis of genomic data. For example, functions are available for all but the internal Roche preprocessing methods.

Typically, the gene expression values are computed by first reading the CEL files of interest into an `AffyBatch` object, say `ab`, by

```
> library(affy)
> cels <- list.celfiles(path, full = TRUE)
> ab <- read.affybatch(filenames = cels)
```

where `path` is the name of the directory in which the CEL files are stored, and `cels` is a character vector naming the CEL files with their `full` directory path. Afterwards, the probe intensities stored in `ab` are preprocessed by calling the corresponding `R` function. The PLM signals, e.g., can be generated using either the function `threestep` or by

```
> library(affyPLM)
> out <- fitPLM(ab)
> signal.plm <- pset2eset(out)
```

TABLE 5.1. Run times in seconds for the applications of, on the one hand, `ReadAffy` and `fitPLM`, and on the other hand, `just.rmaplm` to different numbers of Affymetrix HG-U133_Plus_2 chips on an AMD Athlon XP 3000+ machine with 1 GB of RAM.

	10	20	25	30	50	70	90	100
`ReadAffy/fitPLM`	166	348	`Error`	`Error`	–	–	–	–
`just.rmaplm`	130	226	251	318	530	779	1,133	`Error`

As Table 5.1 shows, this approach works only for a small number of microarrays, since the construction of an `AffyBatch` object is very memory-consuming.

In a CRCA project of Roche Diagnostics, however, the PLM signals of more than 400 Affymetrix HG-U133_Plus_2 chips have to be computed under Windows XP on a machine with 4 GB of RAM. Since this thesis is written on an AMD Athlon XP 3000+ machine with 1 GB of RAM, this task is extended to generating the PLM signals of 500 HG-U133_Plus_2 chips on this computer in a reasonable amount of time.

5.2 just-Versions of Preprocessing Algorithms

A first solution to this problem is to avoid the construction of an `AffyBatch` object that contains the intensities of all of the probes, and to read *just* the PM intensities required for the computation of the signals into a matrix. Such just-versions already exist for RMA and GCRMA. For example, RMA signals can be generated by

```
> signal.rma <- just.rma(filenames = cels)
```

which requires much less RAM than

```
> ab <- read.affybatch(filenames = cels)
> signal.rma <- rma(ab)
```

In `just.rma`, the function `read.probematrix` is called to build an $n_{\text{PM}} \times n$ matrix comprising all n_{PM} PM intensities of each of the n samples. Furthermore, an empty `AffyBatch` object is constructed. Even though this object does not contain any data, it can be employed to obtain required array-specific information such as the total number of probe sets and the names of the probes that assign the PMs to the probe sets. Given both the PM matrix and this information, the RMA signals can be computed using the C-code included in `rma`.

We adopt this approach to modify `fitPLM`, and combine this just-version of `fitPLM` with `just.gcrma` to `just.rmaplm` (see Appendix C.3).

As Table 5.1 reveals, the just-version of `fitPLM` enables us to preprocess many more chips than the original version. It is, however, still not possible to determine the PLM signals in a study comprising hundreds of microarrays.

5.3 Preprocessing in Huge Microarray Experiments

The basic idea behind our approach for the joint preprocessing of hundreds of microarrays is not to consider the whole PM matrix at once, but subsets of the data in each of the steps of PLM. Since `R` cannot keep all the subsets in memory, they are repeatedly stored in files and read into `R`. In the following, this procedure implemented in the `R` function `startPLM` with arguments

```
> args(startPLM)
function (filenames, folder = dirname(filenames)[1], mat.xy = NULL,
  batch.size = 1, chunk.size = 100, type.save = c("probeset",
  "both"), qn.save = 5, digits = 12, max.its = 20, asExprs = TRUE,
  save.combine = 100, printDate = TRUE)
```

(see also Appendix C.3) is explained in detail.

Given a vector containing the `filenames` of the CEL files and a `folder` in which temporary files and the final output is stored, the directory specified by `folder` itself (if it does not already exist) and subfolders of `folder` are created.

The CEL files are divided into batches of size `batch.size`, and the probe sets are split into n_{chunk} subsets/chunks each (except for the last one) consisting of `chunk.size` probe sets. A list $\mathcal{L}$ containing n_{chunk} vectors is constructed, where each of the vectors comprises the indices and the names of the rows of the PM matrix corresponding to the probe sets in the respective chunk. All this information along with both the values of the arguments of `startPLM` and the names of the files in which, on the one hand, the background corrected PMs, and on the other hand, the chunks are stored is exported as `RData` files, i.e. as `R` workspaces, into the subfolder utility.

After this preparation step, the actual computation of the PLM signals starts by calling `read.probematrix` to read in and process the CEL files batch-wise. Each column of the current PM matrix is background corrected and then exported as an `RData` file into the subfolder bg. The n_{PM} background corrected PM intensities of this column are sorted, and added to the corresponding values of the vector $\mathbf{q}$, where $\mathbf{q}$ initially consists of n_{PM} zeros and is saved in the subfolder utility after background correcting `qn.save`, $2 \cdot$ `qn.save`, $3 \cdot$ `qn.save`, ... batches.

Subsequent to background correcting the PMs of any of the n CEL files, $\mathbf{q}$ is divided by n to obtain the prototype vector $\mathbf{q}$ for quantile normalization.

In the next step, each vector of background corrected PM intensities is imported individually and quantile normalized as described in Step 2 and 4 of Algorithm 3.3 (with $m = 1$). Afterwards, these PM intensities are split into subsets as predetermined by $\mathcal{L}$, and stored chunk-wise in txt files in the subfolder chunks, where the respective PM intensities of all samples are saved in the same txt file. Denoting the k^{th} entry of $\mathcal{L}$ by $\mathcal{L}_k$, $k = 1, \ldots, n_{\text{chunk}}$, the elements $(j-1)\,|\mathcal{L}_k| + 1, \ldots, j\,|\mathcal{L}_k|$ of the vector stored in the k^{th} chunk file are thus the background corrected and normalized PM intensities of the j^{th} sample, $j = 1, \ldots, n$, in the k^{th} subset.

(For historical reasons, the PM intensities can also be exported chip-wise to the subfolder bgnorm by setting `type.save="both"` in `startPLM`.)

For the summarization step, each of the chunk files is imported and processed separately using the approach described in Section 5.2: A PM matrix is constructed based on probe intensities stored in the currently considered chunk file, the names of the probes in this matrix are obtained from $\mathcal{L}$, and the PLM signals are computed for the probe sets represented in the PM matrix. To ensure that these expression values do not have to be calculated again, if `startPLM` stops because of, e.g., memory problems, they are stored chunk-wise as `RData` files in the subdirectory exprsChunks.

After having summarized all probe sets, the chunk files containing the expression values are successively read in and combined with each other to construct an $m \times n$ matrix $\mathbf{X}$ comprising the signals of all m probe sets and n samples. To safeguard against memory problems, the combined matrix is exported after combining `save.combine`, $2 \cdot$ `save.combine`, ... chunks.

Finally, the gene expression matrix, and if `asExprs = TRUE` in `startPLM` an `exprSet` object containing this matrix are stored in the subfolder `output`, where an `exprSet` object is the typical output of preprocessing functions such as `rma` or `just.rma`. (Note that the class `exprSet` will be replaced by the class `ExpressionSet` in BioConductor 2.0 that will be released in April, 2007, such that the output of `startPLM` will then be an `ExpressionSet` object.)

If the preprocessing unexpectedly fails because of memory (or other) problems, the procedure can be restarted using the function `restartPLM` which only has one argument, namely `folder`. Employing the information stored in `folder`, `restartPLM` performs the same analysis as `startPLM` starting (virtually) at the same point at which the preprocessing has been interrupted.

5.4 Application of `startPLM`

Using an AMD Athlon XP 3000+ machine with 1 GB of RAM, `startPLM` is applied to 500 Affymetrix HG-U133_Plus_2 chips. The whole computation takes less than 10 hours, where the background correction and normalization

requires about 2 hours 43 minutes, the fitting of the PLMs 6 hours 53 minutes, and the fusion of the chunks 19 minutes. During this procedure, about 5.13 GB of files are stored in the directory specified by `folder`, where the final output of `startPLM`, i.e. the `exprSet` object, exhibits a size of 195 MB.

Except for `folder`, the default settings of the arguments of `startPLM` are used in this computation. This in particular means that each chunk file contains at least $11 \cdot 100 \cdot 500 = 550,000$ intensities, since by default `chunk.size = 100` probe sets are considered at once. Lowering `chunk.size` would, on the one hand, increase the number of chunks, but on the other hand, decrease the number of intensities saved in the txt files which might reduce the run time and would make it possible to apply `startPLM` to many more than 500 microarrays.

PART III

High Level Analysis of SNP Data

Chapter 6

Adapting DNA Microarray Methods to SNP Data

6.1 Introduction

An important goal of microarray studies is the construction of a diagnostic chip, i.e. a microarray composed of a small number of genes, enabling to determine if a person has cancer, or which (sub-)type of cancer this patient exhibits. In more statistical terms, this means that a rule for predicting the cancer status of a person based on as few variables as possible should be constructed.

Since the result of the preprocessing is an $m \times n$ matrix $\mathbf{X}$ comprising the expression values of m genes (or more exactly, m probe sets) and n samples/patients, where m is typically in the tens of thousands, the number of genes has to be reduced dramatically. This reduction can, e.g., be done in the following two steps: Firstly, several ten to a few hundred genes are selected using, e.g., multiple testing (see Section 6.3). Secondly, a discrimination method, often Support Vector Machines or Random Forests (see Section 7.3), is applied to this set of variables to further reduce the number of genes using, e.g., a backward elimination approach (e.g., Guyon et al., 2002), or a stepwise selection procedure such as SFFS (**S**equentially **F**loating **F**orward **S**election; Pudil et al., 1994,

Somol et al., 1999), and to construct a classification rule.

This is, of course, a very simplified description of the way from the output of a preprocessing method to a classification rule. In the actual analysis, one also has to consider approaches such as (inner and outer) cross-validation to avoid selection bias. Furthermore, there are other reasonable strategies. For detailed descriptions on how gene expression data might be analyzed, see Gentleman et al. (2005), and for feature extraction in a more general setting, see, e.g., Guyon et al. (2006).

While a large number of papers concerned with high level analyses of gene expression data have been published in recent years, only a few methods dealing with the specific needs of the analysis of SNP data have been proposed. Two of these exceptions are the Multifactor-Dimensionality Reduction (MDR; Ritchie et al., 2001) and logic regression (Ruczinski et al., 2003). In a comparison of these procedures, Rabe (2004) shows that logic regression has several advantages over MDR. Logic regression is, e.g., faster, can handle a larger number of variables, uses a better search strategy, and leads to classification rules that are easier to interpret. Furthermore, in MDR, a new observation can only be classified if this person exhibits a combination of genotypes that has also been observed in the training set. We therefore exclude MDR from our analyses, and take a closer look on logic regression in Chapter 7 and 8.

Since the goals of the analysis of gene expression and genotype data are similar (e.g., identifying genes/SNPs associated with the covariate of interest, classifying patients using genetic markers), one solution to the problem of how to analyze SNP data is to adapt methods developed particularly for the analysis of DNA microarrays to genotype data. These modified approaches can then not only be applied to genotype array data, but also to SNP data measured with, e.g., MALDI-TOF-MS.

In the following sections, this is exemplified by modifying three popular DNA microarray procedures: A method for imputing missing values (Section 6.2), a multiple testing approach (Section 6.3), and a discrimination procedure (Sec-

tion 6.4). The latter two are improved versions of the methods presented in Schwender (2005).

In each of these sections, we propose an algorithm based on matrix algebra that enables the simultaneous computation of all statistics employed by the respective method. These procedures reduce the run time in R substantially in comparison to approaches in which the statistics are determined individually. Since in these algorithms element-wise matrix calculation is used, notations for these computations are introduced in the following definition.

Definition 6.1 (Element-wise Matrix Calculation)
Let $\mathbf{M}$ and $\mathbf{N}$ be two $R \times C$ matrices, and n be a numerical value. Then,

(a) $\mathbf{M} * \mathbf{N}$ is a $R \times C$ matrix with elements $m_{rc} \cdot n_{rc}$, $r = 1, \ldots, R$, $c = 1, \ldots, C$,

(b) $\frac{\mathbf{M}}{\mathbf{N}}$ is a $R \times C$ matrix with entries $\frac{m_{rc}}{n_{rc}}$,

(c) $\mathbf{M} - n$ is a $R \times C$ matrix with elements $m_{rc} - n$.

6.2 Imputation of Missing Values

6.2.1 Missing Values in the GENICA Data Set

Since missing values are a common problem in association studies (Dai et al., 2006), the imputation of missing genotypes is considered as a first example. In the SNP data set of the GENICA study, e.g, about 1.3% of the values are missing (after removing a few women with more than three missing values, see Appendix A.2). A solution to this problem is to only use complete observations, i.e. persons without missing values. This, however, would mean that data of just 63.3% of the women are considered. Moreover, this approach might add bias to the results of the analysis (Greenland and Finkle, 1995).

Therefore, the missing genotypes of the GENICA data set are replaced using the method that performs best in a comparison of already existing approaches for imputing categorical data with a modification of KNNimpute (Troyanskaya

et al., 2003) introduced in Section 6.2.4. Since only 1.3% of the genotypes are missing, we do not consider multiple imputation methods (Little and Rubin, 1987). Since we do not have information on *haplotypes*, i.e. on blocks of SNPs that are inherited together, and thus, are highly correlated, haplotype-based imputation methods such as the one proposed by Dai et al. (2006) are also not included in the comparison.

6.2.2 KNNimpute

Missing expression values should not be a problem anymore when employing Affymetrix microarrays, since nowadays the probe pairs representing a particular gene are distributed over the complete array such that most of the probe intensities can still be used to compute the expression value of the gene even if a whole region of the chip is manually flagged, and thus excluded from subsequent low level analyses because of bad quality. On the first Affymetrix GeneChips, however, all probe pairs composing a probe set were located right beside each other. Therefore, Troyanskaya et al. (2003) have proposed a method called KNNimpute for imputing missing expression values based on weighted k Nearest Neighbors (kNN; Fix and Hodges, 1951).

Let's assume that the expression value x_{ij} of gene i and sample j is missing, and that $\mathcal{L}_k$ is a set comprising the k genes showing the smallest Euclidean distance to gene i and having a value present for the j^{th} sample. Using KNNimpute, x_{ij} is computed by

$$x_{ij} = \sum_{\ell \in \mathcal{L}_k} w_{i\ell} x_{\ell j} \Big/ \sum_{\ell \in \mathcal{L}_k} w_{i\ell}, \tag{6.1}$$

where the weight $w_{i\ell}$ is determined by the reciprocal of the Euclidean distance between gene i and gene ℓ.

For a comparison of the Euclidean distance with other distance measures, and of the weighted mean (6.1) with other estimates of average in the context of KNNimpute, and an application of KNNimpute to protein expression data, see Jung (2006).

6.2.3 Simultaneous Computation of χ^2-Statistics

If k nearest neighbors should be applied to categorical data, a distance measure has to be employed that can cope with this type of data. Such measures are typically based on an $R \times C$ contingency table in which the joint distribution of two variables, say Y and Z, with observation vectors $\mathbf{y}$ and $\mathbf{z}$ each of length n is represented by the numbers

$$n_{rc} = \sum_{j=1}^{n} I\left(y_j = r\right) I\left(z_j = c\right)$$

of observations showing the r^{th} level at Y, $r = 1, \ldots, R$, and the c^{th} level at Z, $c = 1, \ldots, C$.

An example for such a distance measure is

$$d_{\text{Cont}}\big(\mathbf{y}, \mathbf{z}\big) = \sqrt{1 - \text{Cont}^2\big(\mathbf{y}, \mathbf{z}\big)}, \tag{6.2}$$

where Pearson's corrected contingency coefficient

$$\text{Cont}\big(\mathbf{y}, \mathbf{z}\big) = \sqrt{\frac{\min\big\{R, C\big\}}{\min\big\{R, C\big\} - 1} \cdot \frac{\chi^2}{\chi^2 + n}} \tag{6.3}$$

is based on Pearson's χ^2-statistic

$$\chi^2 \;=\; \sum_{r=1}^{R} \sum_{c=1}^{C} \frac{\left(n_{rc} - \tilde{n}_{rc}\right)^2}{\tilde{n}_{rc}} \;=\; \sum_{r=1}^{R} \sum_{c=1}^{C} \frac{n_{rc}^2}{\tilde{n}_{rc}} - n \tag{6.4}$$

for testing the null hypothesis that the two variables are independent by comparing n_{rc} with the numbers

$$\tilde{n}_{rc} = \frac{1}{n} \sum_{c=1}^{C} n_{rc} \sum_{r=1}^{R} n_{rc}$$

expected under the null hypothesis.

As in the analysis of gene expression data, we suppose that $\mathbf{X}$ is an $m \times n$ matrix in which each row represents one of the m variables, i.e. SNPs, and each column one of the n observations. In Algorithm 6.1, it is described how (6.2) can be determined for all $m(m-1)/2$ pairs of the m variables simultaneously under the assumptions that, on the one hand, $C = R$, i.e. all variables exhibit the same number of levels, and on the other hand, none of the values are missing.

Algorithm 6.1 (Pair-wise Contingency Coefficient Based Distances)
Let $\mathbf{X}$ be an $m \times n$ matrix consisting of the values $1, \ldots, R$.

1. Let $\mathbf{X}^{(r)}$ denote an $m \times n$ matrix, $r = 1, \ldots, R$, with elements
$$x_{ij}^{(r)} = \begin{cases} 1, & \text{if } x_{ij} = r \\ 0 & \text{otherwise} \end{cases}.$$

2. For $r, c = 1, \ldots, R$, compute $\mathbf{N}^{(rc)} = \mathbf{X}^{(r)}\mathbf{X}^{(c)\prime}$ and
$$\tilde{\mathbf{N}}^{(rc)} = \frac{1}{n}\mathbf{X}^{(r)}\mathbf{1}_n\mathbf{1}_n'\mathbf{X}^{(c)\prime}. \tag{6.5}$$

3. Calculate
$$\mathbf{Q} = \sum_{r=1}^{R}\sum_{c=1}^{R} \frac{\mathbf{N}^{(rc)} * \mathbf{N}^{(rc)}}{\tilde{\mathbf{N}}^{(rc)}} - n. \tag{6.6}$$

4. The squared distance (6.2) between the i^{th} and the ℓ^{th} variable, $i, \ell = 1, \ldots, m$, represented by the i^{th} and ℓ^{th} row of $\mathbf{X}$ is determined by the $\left(i^{\text{th}}, \ell^{\text{th}}\right)$ element of
$$\mathbf{D}^2_{\text{Cont}} = \mathbf{1}_{m,m} - \frac{R}{R-1} \cdot \frac{\mathbf{Q}}{\mathbf{Q}+n}, \tag{6.7}$$
where $\mathbf{1}_{m,m}$ is a $m \times m$ matrix composed of only ones.

Algorithm 6.1 can be extended to the case $C \neq R$ by setting each element $\tilde{n}_{k\ell}^{(rc)}$ of $\tilde{\mathbf{N}}^{(rc)}$ to $\max\left\{1, \tilde{n}_{k\ell}^{(rc)}\right\}$, and by replacing R in (6.7) by a matrix containing the respective values of $\min\left\{R, C\right\}$. In the next section, it is shown how this algorithm can be modified for data with missing values.

The matrices $\mathbf{N}^{(rc)}$ containing the numbers n_{rc} for all $m(m-1)/2$ pairs of variables can also be used to compute other similarity measures such as the simple or the flexible matching coefficient (Selinski and Ickstadt, 2005). In this thesis, however, only Pearson's corrected contingency coefficient is considered, since this measure seems to be better suited for identifying groups of (highly) correlated SNPs (cf. Müller et al., 2005).

6.2.4 KNNimpute for Categorical Data

Even though Algorithm 6.1 will still work if there are missing values, the resulting distances will not be correct, since the number of observations showing no missing value at a particular variable might differ from the total number n of observations. To solve this problem, let $\mathbf{X}^{\mathrm{NA}}$ be an $m \times n$ matrix with

$$x_{ij}^{\mathrm{NA}} = \begin{cases} 1, & \text{if } x_{ij} \text{ is not missing} \\ 0, & \text{if } x_{ij} \text{ is missing} \end{cases},$$

and replace n in (6.5) – (6.7) by the $m \times m$ matrix $\mathbf{N} = \mathbf{X}^{\mathrm{NA}} \left(\mathbf{X}^{\mathrm{NA}}\right)'$.

Since the row-wise sums of $\mathbf{X}^{(r)}$ in (6.5) only take individual but not pairwise missing values, i.e. missing values appearing in either of the two considered variables, into account, it is additionally necessary to replace them by $\mathbf{Z}^{(r)} = \mathbf{X}^{(r)} \left(\mathbf{X}^{\mathrm{NA}}\right)'$ such that (6.5) becomes

$$\tilde{\mathbf{N}}^{(rc)} = \frac{\mathbf{Z}^{(r)} * \left(\mathbf{Z}^{(c)}\right)'}{\mathbf{N}}.$$

The so modified Algorithm 6.1 can also be applied to a matrix $\mathbf{X}$ in which some of the values are missing. In Table 6.1, the run time of Algorithm 6.1 in `R` is compared with the time required to compute the distances individually.

TABLE 6.1. Comparison of the run times (in seconds) of Algorithm 6.1 and the individual calculation of the distance d_{Cont} for each pair of m categorical variables, where each variable exhibits $c = 3$ levels, and the number of observations is $n = 1,000$.

m	Algorithm 6.1	Individual
10	0.01	0.15
50	0.07	4.25
100	0.33	17.32
200	1.22	70.06
500	7.79	474.22

This table shows that Algorithm 6.1 is substantially faster, in particular, if the number of variables is large.

After computing the pair-wise distances between the variables using the modified Algorithm 6.1, a missing value x_{ij} can be imputed similar to the original version of KNNimpute by identifying the set $\mathcal{L}_k$ comprising the k variables showing the smallest distances d_{Cont} to variable i and having a value present for the j^{th} observation, and by determining x_{ij} by weighted majority voting, i.e. by

$$x_{ij} = \underset{r=1,\ldots,R}{\arg\max} \sum_{\ell \in \mathcal{L}_k} d_{\text{Cont}}^{-1}\left(\mathbf{x}_{\ell\cdot}, \mathbf{x}_{i\cdot}\right) I\left(x_{\ell j} = r\right), \tag{6.8}$$

where $\mathbf{x}_{\ell\cdot} = \begin{bmatrix} x_{\ell 1} & \ldots & x_{\ell n} \end{bmatrix}'$ and $\mathbf{x}_{i\cdot} = \begin{bmatrix} x_{i1} & \ldots & x_{in} \end{bmatrix}'$. (In (6.8), the normalization factor $\sum_{\ell \in \mathcal{L}_k} d_{\text{Cont}}^{-1}(\mathbf{x}_{\ell\cdot}, \mathbf{x}_{i\cdot})$ is omitted, since (6.8) is not affected by this constant.)

6.2.5 Imputing Missing Values of the GENICA Data Set

Contrary to the discrimination approach of k Nearest Neighbors (cf., e.g., Ripley, 1996) in which the k nearest observations are employed to predict the class of a new observation, Troyanskaya et al. (2003) borrow strength from the huge number of genes by imputing the missing values based on the k nearest genes.

In the GENICA data set, however, the number of observations is much larger than the number of SNPs. The approach proposed in Section 6.2.4 is therefore not only applied to $\mathbf{X}$, but also to $\mathbf{X}'$ to figure out which of these approaches works better for the GENICA data. Furthermore, it is examined if the weighted majority voting (6.8) performs better than an unweighted voting.

In these comparisons, only the genotypes of the 759 women with no missing value are employed. After removing 1%, 2%, 5%, or 10% of the genotypes randomly, the missing values are replaced using KNNimpute for categorical data, and the imputed values are compared with the real genotypes.

In Figure 6.1, the fractions of falsely imputed genotypes for different values of k are displayed. (Since the plots of all four cases look similar, only the figures

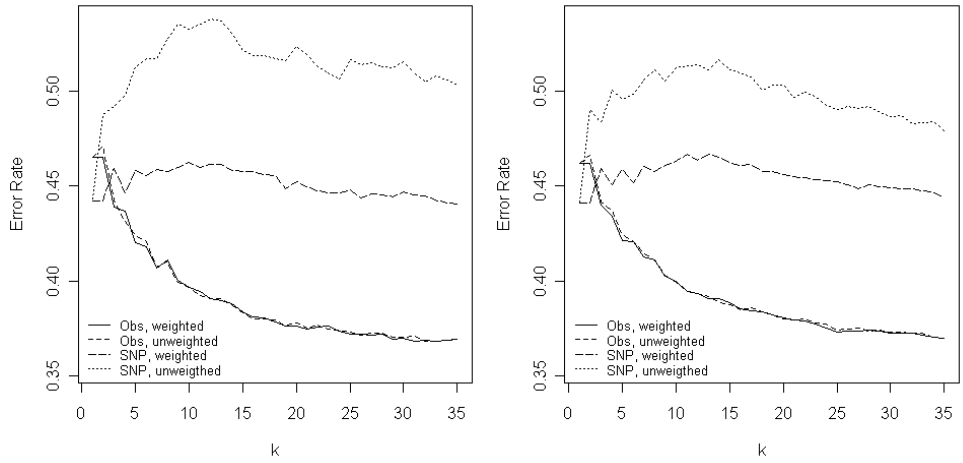

FIGURE 6.1. Fractions of falsely imputed values if (weighted) k nearest neighbors is applied to either the observations or the SNPs to impute the 1% (left panel) or 2% (right panel) artificially generated missing values in the GENICA data set.

for the imputation of 1% and 2% missing values are shown. The other two cases are displayed in Figure B.5 in Appendix B.1.) Figure 6.1 reveals that considering the k nearest observations results in a lower fraction of falsely imputed values than employing the k nearest SNPs. Only for $k < 3$, the latter approach shows a smaller error rate which might be due to the fact that in this case mostly SNPs from the same gene are used to impute the missing genotypes (cf. Müller et al., 2005).

Even though weighting the votes of the k nearest observations does not improve the imputation, (6.8) is used in a comparison of KNNimpute with other approaches. In this comparison, the removed genotypes of the complete observations from the GENICA data set are imputed by the SNP-wise mode, i.e. typically by the homozygous reference genotype, by a random draw from the distribution of the respective SNP, or by a random draw from the conditional distribution of the SNP given the case-control status. Besides those three ad-hoc approaches, two more sophisticated imputation methods are also applied to the data sets.

In the first procedure, the missing values are initially replaced by the mode of

the respective variable. Afterwards, Random Forests is applied to this data set (for a description of Random Forests, see Section 7.3.3), and the proximity, i.e. the fraction of trees containing two particular observations in the same terminal node, is computed for each pair of observations. The missing genotypes are then recalculated by weighted majority voting, where the votes resulting from the trees are weighted by the proximities. This procedure is repeated several times, and the values determined in the final iteration are the estimates of the missing genotypes.

The second approach proposed by Dai et al. (2006) is based on a combination of Gibbs sampling (see, e.g., Gelman et al., 2003) and CART (cf. Section 7.3.2). Gibbs sampling is used to iteratively sample from the conditional distribution of the missing data of the j^{th} observation given the values computed for the missing genotypes of the other observations in the previous steps of Gibbs sampling and the complete data of all observations, whereas CART is employed to model this full conditional distribution (for more details, see Dai et al., 2006).

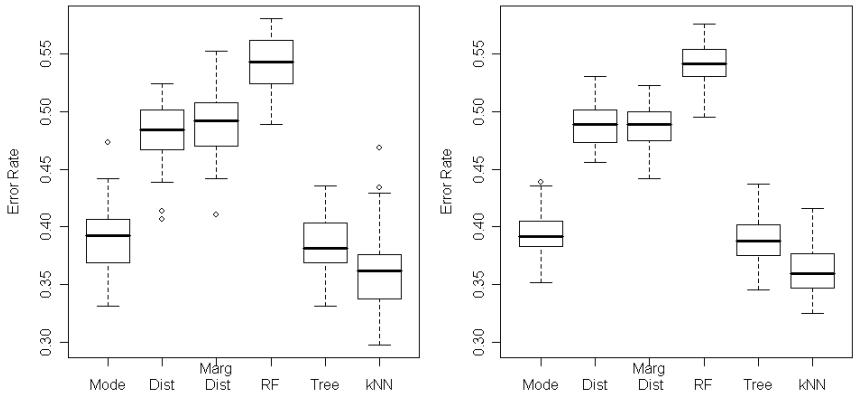

FIGURE 6.2. Fractions of falsely imputed genotypes when replacing the 1% (left panel) or 2% (right panel) missing values by the mode, by a draw from the SNP-wise distribution, by a draw from the conditional distribution of the SNP given the case-control status, by the Random Forests based method (5 Iterations, 500 trees with 6 SNPs at each node), by the procedure of Dai et al. (2006) with one iteration, and by KNNimpute for categorical data ($k = 50$).

For the comparison, these two methods and KNNimpute are optimized by considering different values for their parameters. In KNNimpute, the number of observations is set to $k = 1, \ldots, 60$; in the CART based approach, 1, 2, 5, 10 (recommended by Dai et al., 2006), 15 and 20 iterations are used; and in the Random Forests based method, on the one hand, one to six iterations, and on the other hand, 500, 1000 and 2000 trees with $\lfloor\sqrt{39}\rfloor = 6$, 3, 12 and 39 randomly selected SNPs at each node are considered.

In Figure 6.2, the fractions of falsely imputed genotypes of the applications of these procedures with optimized parameters are displayed. (Again, only the cases with 1% and 2% removed genotypes are presented, whereas both the 5% and the 10% case are displayed in Figure B.6.) This figure reveals that KNNimpute leads to a slightly lower error rate than the approach of Dai et al. (2006) and the imputations based on the mode which in turn exhibit lower error rates than the other procedures.

Therefore, the 1.3% missing genotypes in the GENICA data set are imputed using KNNimpute with $k = 50$ nearest observations, where this approach is implemented in the function `replace.by.wknn` (see Appendix C.3). The resulting data set is considered in the applications of the following sections.

6.3 Significance Analysis of Microarrays

6.3.1 Multiple Testing

An important task in DNA microarray experiments is the identification of differentially expressed genes, i.e. of genes showing expression values that differ substantially between several groups or under several conditions. Detecting such genes requires methods that can cope with this huge multiple testing problem in which tens of thousands of hypotheses are tested simultaneously.

Naturally, the value of a statistic appropriate for testing if the expression values are associated with the covariate of interest and the corresponding p-value

are computed for each gene. Afterwards, these raw p-values are adjusted for multiplicity such that a Type I error rate is strongly controlled at a prespecified level of significance α. (Alternatively, the level of significance can be adjusted for multiplicity, and the raw p-values are compared with this adjusted level.)

The classical example for a Type I error rate is the family-wise error rate

$$\text{FWER} = \text{Prob}(V \geq 1),$$

where V is the number of false positives, i.e. the number of rejected null hypotheses that are actually true (or in biological terms, the number of genes called differentially expressed by the procedure that are actually not differentially expressed). This error rate is strongly controlled at the level α so that $\text{Prob}(V \geq 1) \leq \alpha$ by approaches such as the Bonferroni correction or the procedures of Westfall and Young (1993). An overview on such methods is given by Shaffer (1995). Dudoit et al. (2003) compare procedures for controlling this and other error rates in their application to DNA microarray data.

In the classical multiple testing situation in which rarely more than 20 hypotheses are tested simultaneously, it is reasonable to keep down the probability of one or more false positives. In the analysis of microarray data, however, thousands of genes are considered simultaneously. Moreover, a few false positives are acceptable in such experiments as long as their number is small in proportion to the total number R of identified genes, i.e. rejected null hypothesis. In this situation, the FWER might be too conservative. Hence, another error rate, namely the false discovery rate

$$\text{FDR} = \begin{cases} \text{E}\left(V/R\right), & \text{if } R > 0 \\ 0, & \text{if } R = 0 \end{cases}$$

proposed by Benjamini and Hochberg (1995), has become popular in the analysis of microarray data. For a given rejection region Γ, the FDR can be estimated by

$$\widehat{FDR}(\Gamma) = \pi_0 \frac{\text{E}_{\text{H}_0}\Big(\#\{D_i \in \Gamma\}\Big)}{\max\Big\{\#\{d_i \in \Gamma\}, 1\Big\}},$$

where d_i is the value of a test statistic for gene i, $i = 1, \ldots, m$, $\mathrm{E}_{\mathrm{H}_0}\big(\#\{D_i \in \Gamma\}\big)$ is the under the null hypothesis expected number of test statistics falling into Γ, and π_0 is the prior probability that a gene is not differentially expressed (cf. Storey, 2002, and Schwender, 2003). In the following sections, π_0 is estimated by the procedure of Storey and Tibshirani (2003b) presented in Algorithm 6.2.

Algorithm 6.2 (Estimation of $\boldsymbol{\pi_0}$)

Let p_i be the raw p-value of gene i, $i = 1, \ldots, m$.

1. For $\upsilon = 0, 0.01, 0.02, \ldots, 0.95$, compute $\hat{\pi}_0(\upsilon) = \#\big\{p_i > \upsilon\big\}/\big((1-\upsilon)\,m\big)$.
2. Fit a natural cubic spline ncs with three degrees of freedom through the data points $\big(\upsilon,\, \hat{\pi}_0(\upsilon)\big)$.
3. Estimate π_0 by $\hat{\pi}_0 = \min\big\{ncs(1), 1\big\}$.

It, however, can be shown that the FDR is actually too liberal (see the discussion of Ge et al., 2003). Another drawback of the FDR is that this error rate controls the proportion V/R of false positives among the rejected null hypothesis only on average. Therefore, Genovese and Wasserman (2002) consider the tail probability for the proportion of false positives

$$\mathrm{TPPFP}(\gamma) = \mathrm{Prob}\big(V/R \leq \gamma\big)$$

for a prespecified fraction $0 < \gamma < 1$. An overview on procedures controlling the TPPFP is given by van der Laan et al. (2004), whereas van der Laan et al. (2005) introduce a new controlling method based on bootstrap and an empirical Bayes approach, and show that this procedure outperforms other approaches for controlling the TPPFP.

Apart from adjusting p-values, there also exist other approaches based on, e.g., QQ plots or the empirical Bayes framework that can be used to adjust for multiplicity (for the latter, see Efron et al., 2001). If the observed test statistics

are plotted against the under the null hypothesis expected values most of these points will approximately lie on the diagonal. The points that differ substantially from this line correspond to genes that are most likely differentially expressed. The Significance Analysis of Microarrays (SAM; Tusher et al., 2001) described in the following section can be used to specify what "differs substantially" means.

6.3.2 SAM Procedure

For each gene i, $i = 1, \ldots, m$, Tusher et al. (2001) compute the moderated test statistic

$$d_i = \frac{r_i}{s_i + s_0}, \tag{6.9}$$

where the fudge factor s_0 is added to the denominator of an ordinary statistic $t_i = r_i/s_i$ appropriate for testing if the expression values are associated with the response to prevent that genes with very low expression values dominate the results of the analysis (for details on s_0, see Tusher et al., 2001, and for its computation, Schwender, 2003). For example, in the two-class case, t_i is the ordinary t-statistic.

In their empirical Bayes approach, Efron et al. (2001) also use (6.9), but compute the value of s_0 in a different way, whereas Smyth (2004) proposes a different moderated t-statistic based on a hierarchical model suggested by Lönnstedt and Speed (2002).

Since the null distribution of (6.9) is typically unknown, it is estimated by a permutation based approach. In Schwender (2003), we show how the SAM procedure can be modified if the null distribution is known. In Algorithm 6.3, the Significance Analysis of Microarrays is outlined for both cases.

Algorithm 6.3 (Significance Analysis of Microarrays)

Let $\mathbf{X}$ be an $m \times n$ matrix comprising the expression values of m genes and n observations, $\mathbf{y}$ be the response vector of length n, B be the number of permutations, and $\mathcal{D}$ be a set of strictly positive thresholds Δ.

1. For each gene i, $i = 1, \ldots, m$, compute the value d_i of a statistic appropriate for testing if its expression values are associated with the response.

2. If the null distribution is known, determine the expected test scores $d^0_{(i)}$ by the $(i - 0.5)/m$ quantile of this distribution. Otherwise, assess $d^0_{(i)}$ by computing the m permuted d-statistics d_{ib} for each permutation b, $b = 1, \ldots, B$, of the n values of the response, and by setting $d^0_{(i)} = \sum_{b=1}^{B} d_{(i)b}/B$.

3. For Δ in $\mathcal{D}$

   (a) compute

$$\mathrm{cut}_{\mathrm{up}}(\Delta) = \begin{cases} d_{(i_1)}, & \text{if } i_1 = \min\limits_{i \geq i_0} \Big\{ i : d_{(i)} - d^0_{(i)} \geq \Delta \Big\} \text{ exists} \\ \infty & \text{otherwise} \end{cases},$$

   where $i_0 = \sum_{i=1}^{m} I\Big(d^0_{(i)} < 0\Big) + 1$, and

$$\mathrm{cut}_{\mathrm{low}}(\Delta) = \begin{cases} d_{(i_2)}, & \text{if } i_2 = \max\limits_{i < i_0} \Big\{ i : d_{(i)} - d^0_{(i)} \leq -\Delta \Big\} \text{ exists} \\ -\infty & \text{otherwise} \end{cases},$$

   (b) let $\mathcal{S}_\Delta = \Big\{ i : \, d_i \notin \Gamma^C_\Delta = \big(\mathrm{cut}_{\mathrm{low}}(\Delta), \mathrm{cut}_{\mathrm{up}}(\Delta)\big) \Big\}$ be the set of genes called differentially expressed,

   (c) estimate the FDR for $\mathcal{S}_\Delta$ by

$$\widehat{\mathrm{FDR}}(\Delta) = \frac{\pi_0 a m}{\max\Big\{ \big|\mathcal{S}_\Delta\big|, 1 \Big\}}, \tag{6.10}$$

   where

$$a = \begin{cases} 1 - \displaystyle\int_{\mathrm{cut}_{\mathrm{low}}(\Delta)}^{\mathrm{cut}_{\mathrm{up}}(\Delta)} f_0(z)\, \mathrm{d}z & \text{if the null density } f_0 \text{ is known} \\ \dfrac{1}{mB} \displaystyle\sum_{b=1}^{B} \sum_{i=1}^{m} I\Big(d_{ib} \notin \Gamma^C_\Delta\Big) & \text{otherwise} \end{cases}.$$

In this algorithm implemented in the BioConductor package `siggenes` (see Appendix C.1, and Schwender et al., 2006a), several values for the threshold Δ are considered. Afterwards, the value of Δ is chosen that provides the best balance between the number of identified genes and the estimated FDR, i.e. that allows to simultaneously achieve the two competing goals "As many genes as possible" and "As low FDR as possible" as well as possible (see Section 6.3.4 for an example of how Δ can be chosen).

6.3.3 SAM for Categorical Data

A statistic appropriate for testing if the distribution of a categorical variable with C levels differs between R groups is Pearson's χ^2-statistic (6.4). Since the small denominator problem does not show up in this case, it is not necessary to add the fudge factor s_0 to (6.4). Therefore, SAM can be applied to SNPs – and to any other type of categorical data (cf. Stange et al., 2006) – by setting $d_i = \chi_i^2$, $i = 1, \ldots, m$. Since in SAM it is assumed that all variables follow the same null distribution, the number of levels the variable can take must be same for each of the m variables.

Similar to Section 6.2.3, (6.4) has to be computed for tens of thousands of SNPs. In Algorithm 6.4, a procedure based on matrix calculation is presented enabling the simultaneous determination of a huge number of χ^2-statistics which reduces the run time of individual computations substantially (see Table B.1).

Algorithm 6.4 (Row-wise Pearson's χ^2-Statistic)

Let $\mathbf{X}$ be an $m \times n$ matrix in which each row corresponds to a categorical variable exhibiting the values $1, \ldots, C$, and $\mathbf{y}$ be a vector comprising the class labels $1, \ldots, R$ of the n observations represented by the columns of $\mathbf{X}$.

1. Let $\mathbf{X}^{(c)}$ denote an $m \times n$ matrix with elements $x_{ij}^{(c)} = I\big(x_{ij} = c\big)$, $c = 1, \ldots, C$, and $\mathbf{Y}$ an $n \times R$ matrix with entries $y_{jr} = I\big(y_j = r\big)$.

2. For $c = 1, \ldots, C$, set $\mathbf{Y}^{(c)} = \mathbf{X}^{(c)}\mathbf{Y}$ and $\tilde{\mathbf{Y}}^{(c)} = n^{-1}\mathbf{X}^{(c)}\mathbf{1}_n\mathbf{1}_n'\mathbf{Y}$.

3. The vector $\mathbf{d}$ comprising Pearson's χ^2-statistics for testing each variable represented in $\mathbf{X}$ if its distribution differs between the groups specified by $\mathbf{y}$ is given by

$$\mathbf{d} = \sum_{c=1}^{C} \frac{\mathbf{Y}^{(c)} * \mathbf{Y}^{(c)}}{\tilde{\mathbf{Y}}^{(c)}}\mathbf{1}_R - n. \tag{6.11}$$

After calculating the observed d-values, the expected $d_{(i)}^0$-values can be determined by the $(i-0.5)/m$ quantiles of the $\chi^2_{(R-1)(C-1)}$-distribution, $i = 1, \ldots, m$. However, if the assumptions for the approximation to the χ^2-distribution are not met (see, e.g., Büning and Trenkler, 1994, p. 224), $d_{(i)}^0$ is computed by averaging over $d_{(i)b}$, $b = 1, \ldots, B$, where all mB d_{ib}-values can be assessed simultaneously by considering the $B \times n$ matrix $\mathbf{L}$ in which each row corresponds to one of the B permutations of the class labels comprised by $\mathbf{y}$. If $\mathbf{L}^{(r)}$, $r = 1, \ldots, R$, is defined analogous to $X^{(c)}$, then the matrix $\mathbf{D}^{\text{Perm}} = \{d_{ib}\}$ can be determined by

$$\mathbf{D}^{\text{Perm}} = \sum_{c=1}^{C}\sum_{r=1}^{R} \frac{\left(\mathbf{X}^{(c)}\mathbf{L}^{(r)\prime}\right) * \left(\mathbf{X}^{(c)}\mathbf{L}^{(r)\prime}\right)}{\tilde{\mathbf{y}}_r^{(c)} \otimes \mathbf{1}_B'} - n,$$

where $\tilde{\mathbf{y}}_r^{(c)}$ is the r^{th} column of $\tilde{\mathbf{Y}}^{(c)}$, and $\otimes$ is the symbol for the Kronecker product. Finally, the vector $\mathbf{d}^0$ containing the m $d_{(i)}^0$-values is given by

$$\mathbf{d}^0 = \mathbf{D}^{\text{sort}}\mathbf{1}_B,$$

where the elements of $\mathbf{D}^{\text{sort}}$ are $d_{ib}^{\text{sort}} = d_{(i)b}$.

6.3.4 Application to SNP Data

To identify the SNPs of the HapMap data set (see Appendix A.3) showing a distribution that differs substantially between the 45 Han Chinese from Beijing and the 45 Japanese from Tokio, SAM for categorical data can be applied to these genotype data set using the `R` function `sam` (see Appendix C.1). The analysis of the $m = 121,774$ SNPs requires a run time of 14.81 seconds (from which

about 8 seconds are due to checking for, e.g., incorrect data) if the $d^0_{(i)}$-values are computed by the quantiles of the χ^2_2-distribution, whereas it takes about 183 seconds if these expected scores are determined using $B = 100$ permutations.

By default, `sam` computes the number of identified SNPs and the estimated FDR for ten values of Δ equidistantly spaced between 0.1 and $\max_i \left| d_{(i)} - d^0_{(i)} \right|$. The output of the permutation based SAM analysis of the HapMap data set `hapmap` is thus given by

```
> cl <- rep(1:2, 45)  # class labels
> sam.out <- sam(hapmap, cl, method = cat.stat)
> sam.out
SAM Analysis for Categorical Data

   Delta     p0      False  Called     FDR
1    0.1  0.787  105189.85  108582  0.7626
2    0.9  0.787   15476.45   25028  0.4868
3    1.6  0.787    3006.52    7312  0.3237
4    2.4  0.787     639.51    2413  0.2086
5    3.2  0.787     127.96     764  0.1318
6    3.9  0.787       23.2     211  0.0866
7    4.7  0.787       4.05      61  0.0523
8    5.5  0.787       0.82      19  0.0340
9    6.2  0.787       0.23      14  0.0129
10   7.0  0.787          0       1       0
```

where `p0` is the natural cubic spline based estimate for π_0 (see Algorithm 6.2), `False` $= am$, cf. (6.10), `Called` is the number of identified SNPs, and `FDR = p0 * False / Called` is the estimated FDR.

These statistics can be obtained for other values of Δ using the `R` function `print`. Let's say we aim to identify about 200 SNPs, and to control the FDR at a level of about 5%. In this case, we would take a closer look on the Δ-values between 3.9 and 4.7.

```
> print(sam.out, seq(3.9, 4.7, 0.2))
SAM Analysis for Categorical Data
```

```
   Delta     p0  False  Called     FDR
1    3.9  0.787  23.20     211  0.0866
2    4.1  0.787  19.62     189  0.0817
3    4.3  0.787  13.85     157  0.0694
4    4.5  0.787  10.34     127  0.0641
5    4.7  0.787   4.05      61  0.0523
```

Having selected a reasonable value of Δ, say $\Delta = 4.3$, the SAM plot, i.e. the plot of $d_{(i)}$ vs. $d^0_{(i)}$, $i = 1, \ldots, m$, can be generated using the function `plot` (see Figure 6.3), and information on the identified SNPs such as their names, their d-values, and their q-values (see Storey and Tibshirani, 2003b) can be obtained by employing the function `summary`, or stored in an html file using `sam2html` (for more details on the features of `sam`, see Schwender et al., 2006a). Since the output of both `summary` and `sam2html` is too long to be displayed here it can be found at http://www.statistik.uni-dortmund.de/de/content/einrichtungen/lehrstuehle/personen/holgers/sam.hapmap.html. Contrary to the output of a SAM analysis of gene expression data, this html file, however, does not contain links to

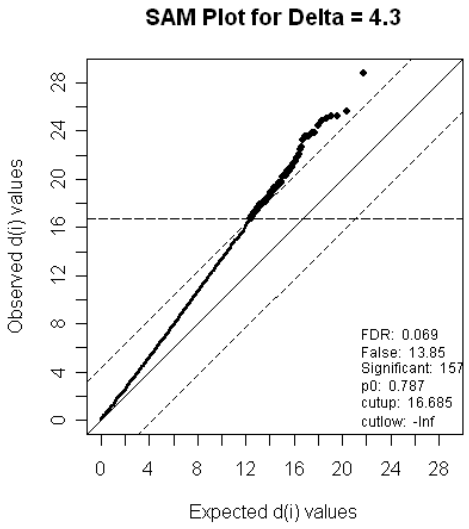

FIGURE 6.3. SAM plot for the HapMap data. Large solid dots mark the identified SNPs, whereas SNPs not detected as significant are represented by small solid dots. While the two dashed diagonal lines have a distance of Δ to the 45°-degree line, the $\text{cut}_{\text{up}}(\Delta)$-value is represented by the dashed horizontal line.

public repositories such as dbSNP (http://www.ncbi.nlm.nih.gov/projects/SNP) that provide biological information on the identified SNPs.

In the applications presented in Chapter 7 and 8, this reduced set of 157 SNPs with an estimated FDR of 6.94% is considered.

Even though SAM has actually been developed for the analysis of high-dimensional data, it can also be applied to data from association studies such as the GENICA study consisting of the genotypes of "only" a few ten SNPs. In the left panel of Figure 6.4, the SAM plot for the analysis of the GENICA data set is shown. This plot reveals that only the distribution of one SNP, namely ERCC2_6540 (refSNP ID: rs1799793), substantially differs between the cases and the controls.

SAM cannot only be used to test individual SNPs, but also to detect interesting interactions of SNPs by considering these interactions as variables. However, only interactions can be analyzed in the same application of SAM that show data for the same number of levels (cf. Section 6.3.3). For example, 552 of the 741 two-way interactions of the SNPs from the GENICA study have data available for each of the nine possible combinations of genotypes. The SAM plot of the analysis of these 552 two-way interactions is displayed in the right panel of

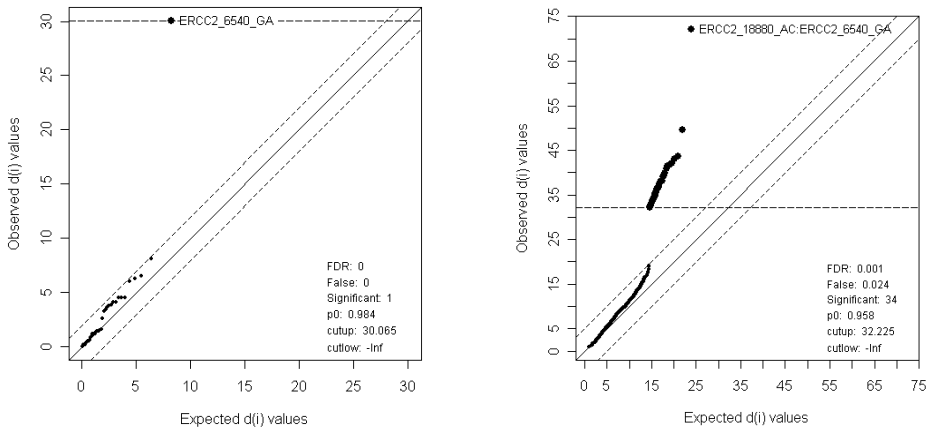

FIGURE 6.4. SAM plots of the analyses of both the individual SNPs (left panel) and the two-way SNP interactions (right panel) from the GENICA data set. The names of the most relevant features are included in the plot using the function `identify`.

Figure 6.4. Each of the 34 identified features is an interaction of ERCC2_6540 with another SNP, where the combination with a second SNP from the gene ERCC2 (**E**xcision **R**epair **C**ross-**C**omplementing group **2**), namely ERCC2_18880 (refSNP ID: rs1052559), that exhibits the seventh largest d-value in the SAM analysis of the individual SNPs shows the largest difference between cases and controls. These results support the findings of Justenhoven et al. (2004).

6.4 Prediction Analysis of Microarrays

6.4.1 Procedure

As a third example, we consider a discrimination method called PAM (**P**rediction **A**nalysis of **M**icroarrays; Tibshirani et al., 2002) that can cope with high-dimensional classification problems. Contrary to other discrimination approaches, it is thus not necessary to reduce the set of genes before applying PAM outlined in Algorithm 6.5 to gene expression data.

Algorithm 6.5 (Prediction Analysis of Microarrays)

Let $\mathbf{X}$ be an $m \times n$ matrix containing the expression data of m genes and n observations, $\mathbf{y}$ be a vector with elements $y_j \in \{1, \ldots, R\}$, $j = 1, \ldots, n$, comprising the classes of the n samples, and $\mathcal{D}$ be a set of strictly positive values.

1. For each gene i, $i = 1, \ldots, m$, compute

   (a) the average expression value $\bar{x}_{ir} = \sum_{j:\, y_j = r} x_{ij}/n_r$ for each class r, $r = 1, \ldots, R$, where n_r is the number of observations in class r,

   (b) the overall centroid $\bar{x}_i = \dfrac{1}{n} \sum_{r=1}^{R} n_r \bar{x}_{ir}$,

   (c) and the test scores

$$d_{ir} = \left(n_r^{-1} + n^{-1}\right)^{-0.5} \cdot \frac{\bar{x}_{ir} - \bar{x}_i}{s_i + s_0}, \quad r = 1, \ldots, R,$$

where $s_i^2 = \dfrac{1}{n-R}\sum_{r=1}^{R}\sum_{j:\,y_j=r}(x_{ij}-\bar{x}_{ir})^2$ is the pooled within-class variance of gene i, and the fudge factor s_0 is estimated by the median over the m standard deviations s_i.

2. For each $\Theta \in \mathcal{D}$, determine

$$d_{ir}^{\Theta} = \operatorname{sign}\big(d_{ir}\big)\big(|d_{ir}| - \Theta\big)_+, \quad i = 1,\ldots,m,\ r = 1,\ldots,R,$$

where $z_+ = z \cdot I\big(z > 0\big)$ and $\operatorname{sign}(z) = (-1)^{I(z<0)}$, and calculate the shrunken centroids

$$\bar{x}_{ir}^{\Theta} = \bar{x}_i + \sqrt{(n_r^{-1} + n^{-1})} \cdot (s_i + s_0) \cdot d_{ir}^{\Theta}. \tag{6.12}$$

Considering an increasing set of values of the shrinkage parameter Θ successively shrinks the d_{ir}-values towards zero. If $d_{ir} = 0$ for all R scores of gene i, then this gene can be removed from the set $\mathcal{S}_\Theta$ of genes used in the classification of a new observation, as it will not contribute anymore to the prediction of the class. Hence, PAM cannot only be employed for discrimination, but also for variable selection.

Let's assume we have identified the value of Θ that minimizes the misclassification error estimated by ten-fold cross validation. Then, the predicted class $\hat{r}$ of a new observation with expression values $\mathbf{x}^* = \begin{bmatrix} x_1^* & \ldots & x_m^* \end{bmatrix}'$ is given by the group r with the nearest shrunken centroid, i.e. by the class r, $r = 1,\ldots,R$, with the minimum discrimination score

$$\delta_r\left(\mathbf{x}^*\right) = \sum_{i \in \mathcal{S}_\Theta} \frac{\big(x_i^* - \bar{x}_{ir}^{\Theta}\big)^2}{\big(s_i + s_0\big)^2} - 2\log\hat{\pi}_r,$$

where $\hat{\pi}_r = n_r/n$ is an estimate of the prior probability π_r of class r. Alternatively, $\hat{r}$ can be determined by the class r with the largest class probability $p_r\left(\mathbf{x}^*\right)$ estimated by

$$\hat{p}_r\left(\mathbf{x}^*\right) = \frac{\exp\big\{-0.5\delta_r\left(\mathbf{x}^*\right)\big\}}{\sum_{\ell=1}^{R}\exp\big\{-0.5\delta_\ell\left(\mathbf{x}^*\right)\big\}}. \tag{6.13}$$

6.4.2 Prediction Analysis of Categorical Data

Contrary to SAM in which just the test statistic has to be changed, PAM cannot directly be adapted to categorical data, since there is no direct counterpart to the average group expression value that can be shrunken towards the overall centroid. Instead of computing a moderated t-statistic, we compare the group-wise distributions of a categorical variable i, $i = 1, \ldots, m$, with its overall distribution, i.e. $n_{rc}^{(i)}$ with $n_{\cdot c}^{(i)}$, $c = 1, \ldots, C$, for each group r, $r = 1, \ldots, R$ (see Table 6.2). To make these numbers comparable, $n_{\cdot c}^{(i)}$ has to be multiplied by $n_{r\cdot}/n$. Since $\tilde{n}_{cr} = n_{\cdot c}^{(i)} n_{r\cdot}/n$, an appropriate score for this situation is Pearson's goodness-of-fit test statistic

$$\chi_{ir}^2 = \sum_{c=1}^{C} \frac{\left(n_{rc}^{(i)} - \tilde{n}_{rc}^{(i)}\right)^2}{\tilde{n}_{rc}^{(i)}} = \sum_{c=1}^{C} \frac{n_{rc}^{(i)} n_{rc}^{(i)}}{\tilde{n}_{rc}^{(i)}} - n_{r\cdot}$$

(cf. Büning and Trenkler, 1994, Chapter 4.2.2). Therefore, PAM can be modified for categorical data by setting $d_{ir} = \chi_{ir}^2$, and by successively lowering d_{ir} towards zero by computing

$$d_{ir}^{\Theta} = \left(d_{ir} - \Theta\right)_+$$

for a set of increasing values of $0 < \Theta < \max_{i,r}\{d_{ir}\}$.

The next step would be to shrink the class distribution towards the overall distribution. This, however, is not as simple as in (6.12), since not a single class prototype is shrunken towards the overall centroid, but C numbers $n_{rc}^{(i)}$ should

TABLE 6.2. Contingency table showing the allocation of n observations into R groups and C levels of a categorical variable i, $i = 1, \ldots, m$.

	1	…	C	$\sum$
Group 1	$n_{11}^{(i)}$	$\cdots$	$n_{1C}^{(i)}$	$n_{1\cdot}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Group R	$n_{R1}^{(i)}$	$\cdots$	$n_{RC}^{(i)}$	$n_{R\cdot}$
$\sum$	$n_{\cdot 1}^{(i)}$	…	$n_{\cdot C}^{(i)}$	n

be shrunken towards the respective $\tilde{n}_{rc}^{(i)}$, $c = 1, \ldots, C$. A solution to this problem is to lower all C distances $\left|\tilde{n}_{rc}^{(i)} - n_{rc}^{(i)}\right|$ by the same amount, i.e. to compute v_{ir} such that

$$n_{rc}^{(i)\Theta} = \tilde{n}_{rc}^{(i)} + v_{ir}\left(n_{rc}^{(i)} - \tilde{n}_{rc}^{(i)}\right) \tag{6.14}$$

for all $c = 1, \ldots, C$, and that

$$\begin{aligned} d_{ir}^{\Theta} &= \sum_{c=1}^{C} \frac{n_{rc}^{(i)\Theta} n_{rc}^{(i)\Theta}}{\tilde{n}_{rc}^{(i)}} - n_{r\cdot} \overset{(6.14)}{=} \sum_{c=1}^{C} \frac{\left(\tilde{n}_{rc}^{(i)} + v_{ir}\left(n_{rc}^{(i)} - \tilde{n}_{rc}^{(i)}\right)\right)^2}{\tilde{n}_{rc}^{(i)}} - n_{r\cdot} \\ &= \sum_{c=1}^{C} \frac{\tilde{n}_{rc}^{(i)2}}{\tilde{n}_{rc}^{(i)}} + v_{ir}^2 \sum_{c=1}^{C} \frac{\left(n_{rc}^{(i)} - \tilde{n}_{rc}^{(i)}\right)^2}{\tilde{n}_{rc}^{(i)}} - 2v_{ir} \sum_{c=1}^{C} \frac{\left(n_{rc}^{(i)} - \tilde{n}_{rc}^{(i)}\right)\tilde{n}_{rc}^{(i)}}{\tilde{n}_{rc}^{(i)}} - n_{r\cdot} \\ &= n_{r\cdot} + v_{ir}^2 \cdot d_{ir} - 2v_{ir} \cdot 0 - n_{r\cdot} = v_{ir}^2 d_{ir}. \end{aligned}$$

Thus, v_{ir} is given by $v_{ir} = \sqrt{d_{ir}^{\Theta}/d_{ir}}$, and

$$\sum_{c=1}^{C} n_{rc}^{(i)\Theta} = \sum_{c=1}^{C} \tilde{n}_{rc}^{(i)} + v_{ir} \sum_{c=1}^{C} \left(n_{rc}^{(i)} - \tilde{n}_{rc}^{(i)}\right) = n_{r\cdot} + v_{ir} \cdot 0 = n_{r\cdot}.$$

Having chosen a value for the shrinkage parameter Θ and specified the set $\mathcal{S}_\Theta$ of categorical variables with at least one non-zero d_{ir}^{Θ}-value, the class of a new observations is predicted by the group r that maximizes the posterior probability

$$p_r\left(\mathbf{x}_\Theta^*\right) = \frac{\pi_r p\left(\mathbf{x}_\Theta^* \mid r\right)}{\sum_{\ell=1}^{R} \pi_\ell p\left(\mathbf{x}_\Theta^* \mid \ell\right)},$$

where $\mathbf{x}_\Theta^*$ denotes the vector comprising the values $x_i \in \{1, \ldots, C\}$ of the variables $i \in \mathcal{S}_\Theta$ for the new observation. In analogy to (6.13), $p\left(\mathbf{x}_\Theta^* \mid r\right)$ is estimated by

$$\hat{p}\left(\mathbf{x}_\Theta^* \mid r\right) = \prod_{i \in \mathcal{S}_\Theta} \hat{p}\left(x_i \mid r\right) = \prod_{i \in \mathcal{S}_\Theta} \frac{n_{rx_i}^{(i)\Theta}}{n_{r\cdot}}.$$

Again, all mR d_{ir}-values can be computed simultaneously using matrix algebra. This can be done in almost the same way as the row-wise Pearson's

χ^2-statistics are calculated in Algorithm 6.4. Only (6.11) has to be replaced by the $m \times R$ matrix

$$\mathbf{D}^{\text{PAM}} = \sum_{c=1}^{C} \frac{\left(\mathbf{Y}^{(c)} - \tilde{\mathbf{Y}}^{(c)}\right) * \left(\mathbf{Y}^{(c)} - \tilde{\mathbf{Y}}^{(c)}\right)}{\tilde{\mathbf{Y}}^{(c)}}.$$

6.4.3 Application to SNP Data

Using the `R` function `pamTheta` (see Appendix C.3), the Prediction Analysis of categorical data is applied to both the HapMap data set comprising all 121,774 SNPs, and the 157 SNPs of this data set preselected by SAM (see Section 6.3.4). In both cases, the misclassification rate (MCR) is estimated for different values of the shrinkage parameter Θ by nine-fold cross-validation, where each of the nine data sets contain five Han Chinese and five Japanese. As shown in the left panel of Figure 6.5, the former application leads to a minimum MCR of 15.6% at $\Theta = 7$. Accepting a little higher MCR, i.e. an MCR of 16.7% (at $\Theta = 8.5$), can reduce the number of SNPs used in the prediction from 362 to 97. Applying PAM just to the preselected SNPs decreases the MCR to 3.3%. A reason for this is that SNPs showing d_{ir}-values larger than Θ, but actually exhibiting group-wise distributions that do not differ substantially from each other, and thus, perturbing the correct classification have been filtered by SAM.

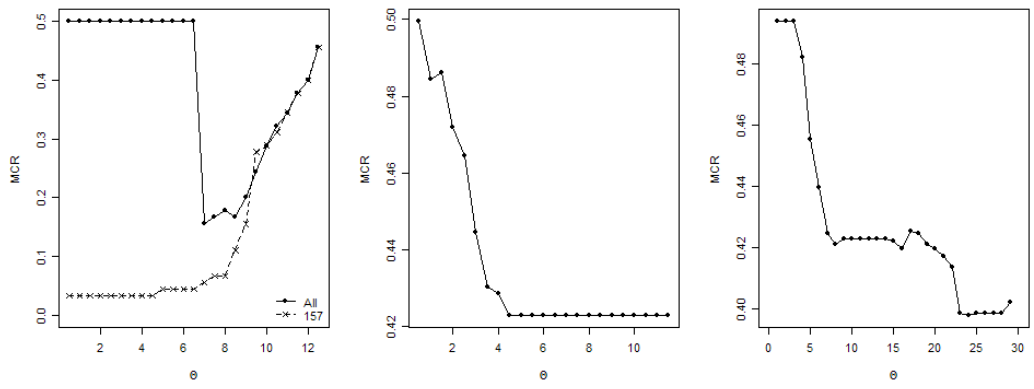

FIGURE 6.5. Misclassification rates for several values of the shrinkage parameter Θ in the application of PAM to the HapMap data (left panel), and to both the SNPs (middle panel) and the two-way interactions (right panel) of the GENICA data set.

If several values of Θ lead to the same MCR, typically the largest value of Θ will be chosen, since this results in the smallest numbers of variables. However, in the analysis of the 157 SNPs, all features are employed in the prediction no matter which value of the shrinkage parameter Θ leading to a MCR of 3.3% is selected. (In- or decreasing the Δ in SAM, and therefore, the number of SNPs will not lead to a smaller MCR.)

As expected from the results of the SAM analyses of the GENICA data set in Section 6.3.4, employing only ERCC2_6540 for class prediction leads to the smallest MCR (42.3%) in the application of PAM to this data set (cf. middle panel of Figure 6.5). This misclassification error estimated by ten-fold cross-validation can be reduced to 39.9% if two-way interactions are analyzed (cf. right panel of Figure 6.5). In this application, again, only the 552 SNP interactions also used in Section 6.3.4 can be considered, since a requirement of PAM is that all categorical variables exhibit the same number of levels. The classification rule leading to the MCR of 39.9% is based on just one interaction, namely the interaction of ERCC2_6540 with ERCC2_18880.

Chapter 7

Comparison of Discrimination Methods Applied to SNP Data

7.1 Introduction

One of the major goals in association studies is the construction of classification rules such as

IF SNP S_1 is of the homozygous reference genotype **AND** SNP S_2 is of the homozygous variant genotype **OR** both SNP S_3 **AND** S_4 are **NOT** of the homozygous reference genotype,

THEN a person has (a higher risk to develop) a particular disease.

A procedure developed for solving exactly this type of problems is logic regression proposed by Ruczinski et al. (2003). In comparison to other discrimination or regression approaches, this method has shown a good performance in its application to SNP data (Kooperberg et al., 2001; Ruczinski et al., 2003, 2004; Witte and Fijal, 2001). In this chapter, we examine if this is also true when considering our data sets. Therefore, logic regression and other discrimination procedures such as Support Vector Machines and Random Forests are applied, on the one hand, to the data from both the HapMap and the GENICA study,

and on the other hand, to the simulated data sets of Simulation 2 (see Appendix A.4). While the other discrimination methods are described briefly in Section 7.3, logic regression is introduced in more details in Section 7.2. The performance of these approaches is then compared in Section 7.4.

7.2 Logic Regression

Logic regression is an adaptive methodology for predicting the outcome in discrimination and regression problems based on Boolean combinations of variables such as

S_{i1} : "SNP S_i is not of the homozygous reference genotype",

S_{i2} : "SNP S_i is of the homozygous variant genotype",

i.e. of binary variables that are either true or false. These variables can be negated by the operator ^{C} (e.g., S_{i2}^C means "SNP S_i is NOT of the homozygous variant genotype"), and combined by the operators $\wedge$ (AND) and $\vee$ (OR) to form a logic expression L that in turn is also either true or false.

In logic regression, such logic expressions are represented by logic trees. For example, the logic tree in the center of Figure 7.1 displays the logic expression $L = S_{11} \wedge S_{21}^C \vee S_{32}$, where the nodes of the tree consist of the operators AND and OR, the terminal nodes or leaves show the variables, and the complement of a variable S, i.e. S^C, is marked by white letters on a black background. For example, in a case-control study in which L is used as a predictor, an individual would be classified as case if L is true, i.e. if either both S_{11} and S_{21} are true, or S_{32} is true, or both $S_{11} \wedge S_{21}$ and S_{32} are true. Otherwise, this person is classified as control.

However, logic trees cannot only be employed as a nice graphical representation of a logic expression, but also to generate new logic trees in the search for the best logic regression model. In Figure 7.1, the permissible moves in this tree-growing process are shown. These moves are

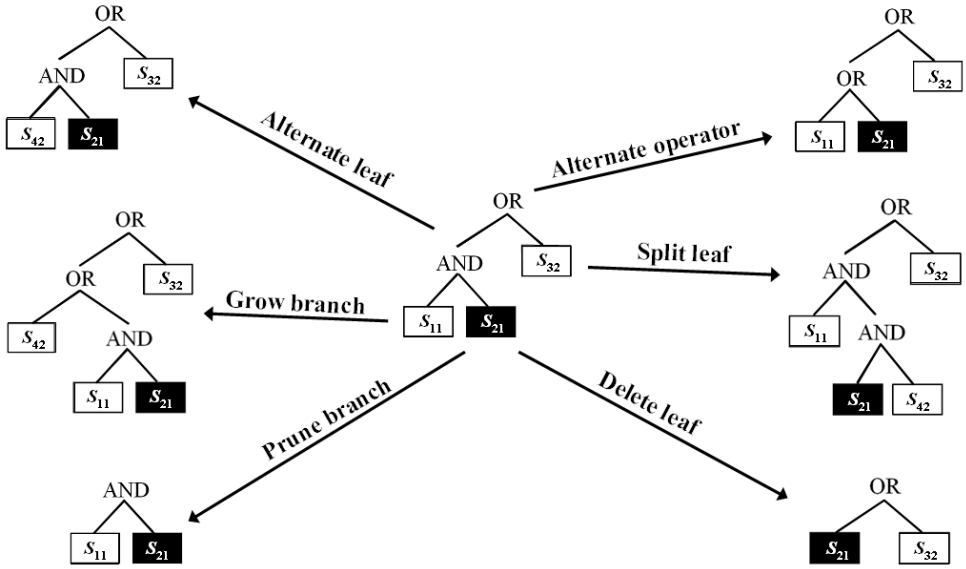

FIGURE 7.1. Move set of logic regression. (*Source*: Ickstadt et al., 2006b, modified version of Figure 4.)

Alternate a leaf: A literal, i.e. a variable or its complement, is replaced by another literal. In the logic tree in the center of Figure 7.1, e.g., S_{21}^C can be replaced by any literal except for S_{11} and S_{11}^C, as these two literals would generate a tautology in the tree.

Alternate an operator: An AND is replaced by an OR, or vice versa.

Grow a branch: At each node (except for the terminal nodes), a new branch can be grown by cutting the subtree starting at this node, adding a new operator to this node, and adding the subtree to the right branch and a leaf to the left branch of this new node.

Prune a branch: This is the countermove to growing a branch in which a branch is removed from the tree.

Split a leaf: Each leaf can be split by replacing it by a subtree consisting of an operator/node, the split leaf and another leaf.

Delete a leaf: This is the countermove to splitting a leaf in which a leaf is removed from the tree.

How a move is selected (e.g., randomly or by optimization), and whether the new tree resulting from this move is accepted depends on the algorithm used in the search for the best model. Since in logic regression simulated annealing introduced by Kirkpatrick et al. (1983) and presented in Algorithm 7.1 is employed as the default search algorithm, it is used in all the applications in this and the following chapter. Note that each move has its countermove, and that using this move set each logic tree can be reached from any other tree in a finite number of moves such that the assumptions of simulated annealing based on the underlying Markov chain theory are met (for details, see Ruczinski, 2000).

Algorithm 7.1 (Simulated Annealing Based Logic Regression)
Given a training set, a cooling scheme, and the number n_{iter} of iterations, a logic regression model is adaptively constructed as follows.

1. Initially, specify L by randomly drawing a logic tree consisting of one leaf.
2. Propose a new tree by randomly selecting a move.
3. Accept the new tree with probability
$$\min\left\{1, \exp\left\{\left(\text{MCR}_L - \text{MCR}_{\text{new}}\right)/t\right\}\right\},$$
where MCR_L and MCR_{new} are the training set errors of L and the new tree, respectively, and the temperature t is specified by the cooling scheme.
4. Repeat Step 2 and 3 n_{iter} times.

In logic regression, a typical cooling scheme would start at $t = 10^{z_{\text{start}}}$, $z_{\text{start}} \in \mathbb{N}$, and lower the temperature to $t = 10^{-z_{\text{end}}}$, $z_{\text{end}} \in \mathbb{N}$, in equal decrements on $\log_{10}$-scale such that in the beginning many different logic regression models are visited, and in the end almost no new tree is accepted if its MCR is larger than the MCR of the current tree L.

Apart from growing a single tree, logic regression provides also the possibility to adaptively grow and combine several logic expressions $L_k, k = 1, \ldots, p$, by a generalized linear model

$$g\big(\mathrm{E}(Y)\big) = \gamma_0 + \sum_{k=1}^{p} \gamma_k L_k \tag{7.1}$$

with response Y, parameters γ_k, $k = 0, \ldots, p$, and link function g. Since our interest centers on binary responses, we assume g to be the logit function. In the following, we refer to the former as single tree approach and to (7.1) as multiple tree approach. (Note that correct notations for these types of logic regression would actually be classification and glm approach, respectively, since $p = 1$ is a possible choice for the number of trees in (7.1). We, however, do not consider this case throughout this thesis.)

In the multiple tree case, a new logic regression model is proposed by randomly picking a move for one of the p trees. Afterwards, this model is fitted and compared with the current model using the deviance

$$-2 \sum_{j=1}^{m} \Big(y_i \ln\left(\hat{\pi}_i\right) + (1 - y_i) \ln\left(1 - \hat{\pi}_i\right) \Big)$$

with $\pi_i = \mathrm{E}(Y_i)$ instead of the MCR (cf. Neter et al., 1996).

In this thesis, we focus on the original version of logic regression that employs the move set displayed in Figure 7.1. Hence, other procedures originated by logic regression such as the full Bayesian logic regression proposed by Fritsch (2006), or the genetic programming based method introduced by Ickstadt et al. (2006a) are not considered.

7.3 Further Discrimination Methods

In this section, the competitors of logic regression in the comparison presented in Section 7.4 are briefly described. A more detailed introduction to most of these methods (all but Random Forests) is provided by Hastie et al. (2001). In Schwender et al. (2004), these discrimination procedures are compared with each other in an application to a first GENICA data set composed of 25 SNPs.

7.3.1 Support Vector Machines

The basic idea of Support Vector Machines (SVM; Vapnik, 2000) is to construct an optimal separating hyperplane between two groups of observations (e.g., cases vs. controls), where optimal means that the distance of the hyperplane to the closest observations from either class, and thus, the margin between these two groups is maximized (cf. Figure 7.2).

In the left panel of Figure 7.2, the two classes are perfectly separated. In this case, the margin and the optimal separating hyperplane bisecting the margin are specified by the observations represented by the solid circles and the solid square lying on the boundary of the margin. These three points are called the support vectors, since only these observations are required to construct the hyperplane, and hence, the classification rule.

If the groups are not perfectly separable, then the data points will be allowed to lie on the wrong side of the margin. In the example displayed in the right panel of Figure 7.2, some of the observations represented by circles can thus be on the right of the left boundary of the margin, and individuals marked

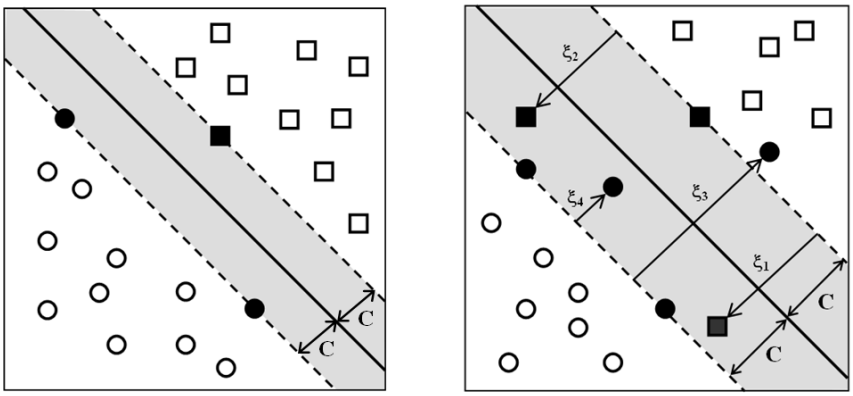

FIGURE 7.2. Support vector classifier for the perfectly separated (left panel) and the non-separable (right panel) case. Circles and squares mark the observations of the two groups. If these symbols are solid, then they represent support vectors. While the separating hyperplane is displayed by the solid line, the shaded region marks the margin. (*Source*: Schwender et al., 2004, modified version of Figure 2.)

by squares can be on the left of the right boundary. Since such data points prevent that a optimal separating hyperplane can be found, they add costs to the discrimination problem. Therefore, these observations are also required to construct the classification rule, and hence, also support vectors.

More formally, we are looking for the hyperplane $\left\{\mathbf{x}:\ \mathbf{x}'\boldsymbol{w}+w_0=0\right\}$ that provides the solution to the optimization problem

$$\min_{\boldsymbol{w},w_0}\left\{0.5\,||\,\boldsymbol{w}||^2+\eta\sum_{j=1}^{n}\xi_j\right\} \tag{7.2}$$
$$\text{subject to}\ \ \xi_j\geq 0,\ \ y_j\left(\mathbf{x}_j'\boldsymbol{w}+w_0\right)\geq 1-\xi_j,\ \ j=1,\ldots,n,$$

where $y_j\in\left\{-1,1\right\}$ and $\mathbf{x}_j\in\mathbb{R}^m$ are the response and the vector containing the values of the m explanatory variables for the j^{th} observation, $j=1,\ldots,n$, $||\,\boldsymbol{w}||^2=\sum_{i=1}^{m}w_i^2\ \left(=C^{-2},\ \text{see Figure 7.2}\right)$, $\xi_j=0$ if the j^{th} observation is on the correct side of the margin, otherwise, $\xi_j>0$ is computed as shown in Figure 7.2, and η is the tuning or cost parameter which has to be optimized separately using, e.g., cross-validation.

So far we have only considered linear hyperplanes. However, two groups might be better separated by a non-linear hyperplane that can be constructed by employing the kernel trick: The data are mapped into a feature space $\mathbb{S}$ that is of a higher dimension than the original input space $\mathbb{R}^m$ using a kernel function

$$K\left(\mathbf{x}_j,\mathbf{x}_\ell\right)=\left\langle\mathbf{h}\left(\mathbf{x}_j\right),\mathbf{h}\left(\mathbf{x}_\ell\right)\right\rangle,$$

where $\mathbf{h}:\mathbb{R}^m\rightarrow\mathbb{S}$ is the actual mapping function, and $\langle\mathbf{a},\mathbf{b}\rangle=\sum_k a_k b_k$. In this feature space, a linear hyperplane is constructed as explained above. Afterwards, the data are mapped back into the original input space in which the hyperplane is not linear anymore.

Two popular examples for kernel functions are the p^{th} degree polynomial kernel

$$K_{\text{poly}}\left(\mathbf{x}_j,\mathbf{x}_\ell\right)=\left(1+\left\langle\mathbf{x}_j,\mathbf{x}_\ell\right\rangle\right)^p$$

and the radial kernel

$$K_{\text{radial}}\left(\mathbf{x}_j,\mathbf{x}_\ell\right)=\exp\left\{-\gamma\,||\,\mathbf{x}_j-\mathbf{x}_\ell||^2\right\},\ \ \gamma>0.$$

7.3.2 Classification and Regression Trees

As implied by its name, CART (**C**lassification **A**nd **R**egression **T**rees; Breiman et al., 1984) can be applied to both classification and regression problems. Here, we focus on classification trees that are constructed by recursively partitioning the data into subsets. Starting with the whole training set at the first node of a CART tree, the variable is identified that best splits the data into two subset. This means that the split v is detected that leads to the largest decrease in impurity

$$\Delta\, i(v, \tau) = i(\tau) - p_1 \cdot i\,(\tau_1) - p_2 \cdot i\,(\tau_2)\,, \tag{7.3}$$

where $i(\tau)$ is the impurity of node τ, and p_1 and p_2 are the proportion of observations going into the two descendant nodes τ_1 and τ_2, i.e. into the two subsets. Typically, the Gini index

$$i_{\text{Gini}}(\tau) = \sum_{r=1}^{R} p_\tau(r)\big(1 - p_\tau(r)\big) \tag{7.4}$$

with $p_\tau(r)$ being the proportion of observations at node τ belonging to class r, $r = 1, \ldots, R$, is used as the measure for the node impurity. In the two-class case, (7.4) becomes $i_{\text{Gini}}(\tau) = 2p_\tau(1)\big(1 - p_\tau(1)\big)$.

Having divided the data into two subset, the corresponding nodes are consi-

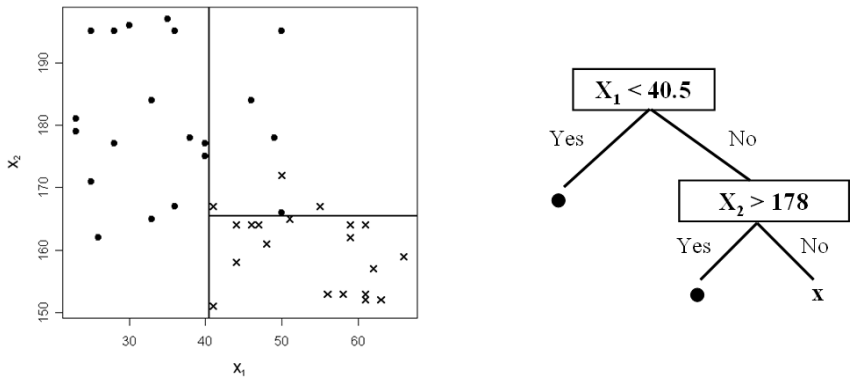

FIGURE 7.3. Two-dimensional feature space partitioned by CART, and the corresponding CART tree. Solid dots represents the observations from group 1, and crosses the individuals from group 2.

dered separately to identify the variable that best splits the data at the respective node into two subsets. This procedure is repeated as long as the number of observations in the subgroups is larger than a prespecified minimum, or the decrease of impurity (7.3) is larger than a prespecified threshold. In Figure 7.3, an example of a recursively partitioned two-dimensional feature space and the corresponding CART tree is displayed.

When an explanatory variable is categorical, all $2^{C-1} - 1$ possible partitions of the C levels of this variable into two subsets have to be considered in the search for the best split. However, if the response is binary this number of partitions can be reduced to C by ordering the classes c, $c = 1, \ldots, C$, of the predictor by the proportion of observations falling into the first response class, and by treating this predictor as an ordinal variable (cf. Breiman et al., 1984, Chapter 4.2.2).

Usually, the constructed tree T_0 is too large, i.e. too specific for the training set. To avoid overfitting, T_0 is pruned using cost-complexity pruning: Let $T \subset T_0$ be any subtree of T_0, and $|\,T|$ be the number of terminal nodes in T. Then, for each value of the tuning parameter ζ, the subtree $T_\zeta \subseteq T_0$ is identified that minimizes the cost complexity criterion

$$C_\zeta(T) = \sum_{\tau=1}^{|\,T|} n_\tau i_T(\tau) + \zeta\,|\,T|\,,$$

where n_τ is the number of observations at the terminal node τ. Afterwards, the final tree $T_{\hat{\zeta}}$ is determined by detecting the value $\hat{\zeta}$ of the tuning parameter ζ that minimizes the misclassification rate estimated by cross-validation (cf. Hastie et al., 2001, p. 270).

7.3.3 Bagging and Random Forests

A major problem of CART trees is their instability: A small change in data can lead to a very different classification rule, and hence, to different predictions for new observations. A solution to this problem is to stabilize this classifier by

employing ensemble methods such as bagging (Breiman, 1996), boosting (Freund and Shapire, 1996; Friedman et al., 2000), or Random Forests (Breiman, 2001).

In bagging outlined in Algorithm 7.2, many CART trees are grown on different subsets of the training data, and a new observation is classified by averaging over the predictions of the individual trees. This is typically done by majority voting, i.e. by assigning the new observation to the class predicted by the majority of the trees.

Algorithm 7.2 (Bagging)

Let n be the number of observations in the training set, and B be the number of iterations.

1. Draw a bootstrap sample of size n from the n observations.
2. Construct a CART tree based on this bootstrap sample.
3. Repeat Step 1 and 2 B times to generate B classification rules.

Even though in Algorithm 7.2 bagging is explicitly applied to CART trees, other discrimination methods can also be used as base classifier in this algorithm. Since CART and logic trees are related – each logic tree can be converted into a CART tree, and vice versa (cf. Appendix A of Ruczinski et al., 2003) – logic trees are also instable classifiers. In Section 7.4, we therefore apply bagging not only to CART, but also to logic regression. To distinguish between these two approaches, the bagging version using logic regression as base learner is called logic bagging in the following.

The last competitor in the comparison presented in the following section is Random Forests which is a further development of bagging. In Random Forests, not only bootstrap samples of the observations are used to construct a large number of CART trees, but also a random subset of variables is considered at each node to identify the best split among this subset, where the subset can differ from node to node.

7.4 Comparison

Since logic regression searches for Boolean combinations of binary variables, each SNP has to be coded by two dummy variables. A biologically meaningful way to do this is to split each SNP S_i into the variables

S_{i1} : "At least one of the bases explaining S_i is the less frequent variant",

S_{i2} : "Both bases explaining S_i are the less frequent variant",

as S_{i1} codes for a dominant variation, and S_{i2} for a recessive mutation. Since SVM also cannot handle categorical data, the same splitting is used in its application to the three data sets. However, the outcomes of $S_{i\ell}$ are not coded by 0 and 1 (or more exactly, false or true), but by -1 and 1.

Both modifications can lead to variables mainly consisting of one value. Thus, all variables showing the same value for more than 97% (95%) of the observations are removed from the GENICA (HapMap) data set leading to 68 (282) binary variables. All other discrimination methods are based on CART, and can therefore be applied to the SNPs themselves.

In the analyses of the data sets, several parameter settings are considered for each of the discrimination procedure except for CART and logic bagging (see Table B.2). In Table 7.2, the misclassification rates (MCRs) of these methods with the optimized parameters summarized in Table 7.1 are displayed. For the reduced HapMap data set consisting of the 157 SNPs preselected by SAM (cf. Section 6.3.4) and the GENICA data set, these misclassification rates are estimated by cross-validation employing the same subsets as in Section 6.4.3, whereas in the applications to the 50 data sets of Simulation 2 that imitate data from real genetic association studies (see Appendix A.4), they are determined by constructing a classification rule on each data set, and applying this rule to another data set such that each data set is used once as training set, and once as test set.

Table 7.2, however, does not contain the MCR of the application of PAM to the two-way SNP interactions of the GENICA data set, since we are here

TABLE 7.1. `R` packages and optimized parameter used in the applications of the discrimination methods to the three data sets leading to the misclassification rates displayed in Table 7.2. (m_t: Number of randomly selected variables at each node.)

Method	`R` package	Parameter	HapMap	GENICA	Simulation 2
		Kernel	Radial	Linear	Radial
SVM	`e1071`	η	0.2	10	2
		γ	10^{-3}	–	10^{-3}
PAM	`c7Tools`	Θ	4.5	11	1
CART	`rpart`	–	–	–	–
Bagging	`ipred`	B	200	100	100
Random	`random`	B	500	1,000	1,000
Forests	`Forest`	m_t	12	12	14
Logic	`LogicReg`	Approach	Single	Single	Single
Regression		n_{leaves}	32	8	8
Logic	`logicFS`	–	Single tree approach with		
Bagging			$n_{\text{leaves}} = 8$, and $B = 100$		

interested in a comparison of methods in which individual variables are used as inputs. Otherwise, PAM – or more precisely, predicting the class of a new observation based on the joint distribution of ERCC2_6540 and ERCC2_18880 – would have shown the smallest MCR. (Similar to PAM, employing also two-way interactions in the analysis would decrease the MCR of CART and Bagging slightly.)

For both the simulated data and the GENICA data set, logic regression exhibits the lowest MCR, where in the former case it comes close to the actual MCR of 32.6%. Using ensemble methods neither reduces the MCR of logic regression nor of CART.

In the analysis of the HapMap data, most of the other discrimination procedures outperform logic regression. This might be due to the fact that in contrast

TABLE 7.2. Misclassification rates for the applications of the discrimination methods to the three data sets with the parameter settings summarized in Table 7.1.

	SVM	PAM	CART	Bagging	Random Forests	Logic Regression	Logic Bagging
HapMap	0	0.033	0.356	0.022	0.011	0.144	0.011
GENICA	0.419	0.423	0.421	0.433	0.428	0.402	0.404
Simulation 2	0.419	0.481	0.371	0.383	0.381	0.342	0.341

to the other two data sets in which five (Simulation 2) or two (GENICA) SNPs have an effect on the case-control status, many SNPs are required to distinguish between the Han Chinese and the Japanese (cf. Section 8.4.2, in particular the left panel of Figure 8.5). However, none of the logic trees contains more than ten variables even if we allow to grow much larger trees. Logic bagging can compensate this problem by considering not just a single logic regression model, but a large number of models each containing a different set of variables. The same applies to CART that performs even worse than logic regression.

Overall, the results of our analyses are similar to the one of, e.g., Kooperberg et al. (2001): Logic regression works well in comparison to other discrimination methods if a few SNP interactions are explanatory for the response (which is also the data situation in Kooperberg et al., 2001). Otherwise, SVM or ensemble methods such as bagging and Random Forests are to be preferred.

Chapter 8

Detection of SNP Interactions Using Logic Regression

8.1 Introduction

Two of the most popular discrimination methods in the analysis of DNA microarrays, SVM (Vapnik, 2000) and Random Forests (Breiman, 2001), cannot only be employed for the classification, but also for variable selection. In RFE-SVM (Guyon et al., 2006), e.g., the squared weights w_i^2, $i = 1, \ldots, m$, cf. (7.2), are used to recursively eliminate features from the set of variables, and to choose the subset leading to the smallest MCR. In Random Forests, the importance of a variable is specified by averaging over the differences between the tree-wise numbers of correctly classified oob (**o**ut-**o**f-**b**ag) observations, i.e. observations that are not in the respective bootstrap sample, when the original or permuted values of this variable are used. Breiman (2001) suggests to apply Random Forests once to the whole data set to select the most important variables, and once to the chosen features to construct the classification rule, whereas Diaz-Uriate and Alvarez de Andres (2006) propose a feature elimination procedure similar to the approach of Guyon et al. (2006).

In Figure 8.1, the values of the importance measure resulting from an appli-

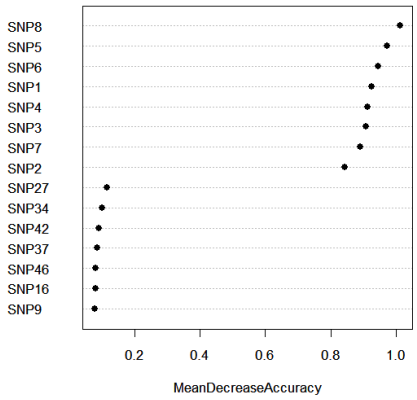

FIGURE 8.1. Importances of the variables of Simulation 1 quantified by Random Forests, where S_i is denoted by SNPi.

cation of Random Forests to the data set of Simulation 1 (see Appendix A.4) are displayed. This figure shows that the eight variables $S_1, \ldots, S_8$ explanatory for the cases are identified as the most important ones, and that all other variables are unimportant. However, Random Forests identifies neither the interactions of interest nor the genotypes explaining the cases. This can be considered as drawbacks for the analysis of data from genetic association studies, as not individual SNPs, but SNP interactions are assumed to be responsible for complex diseases (Garte, 2001; Culverhouse et al., 2002). In Section 8.3, we introduce a procedure called logicFS (logic **F**eature **S**election) based on logic regression that enables the identification of such interactions and the quantification of their importance. In Section 8.4, logicFS is applied to all genotype data sets described in Appendix A.

There already exists an approach based on logic regression for detecting interesting interactions: In MC (**M**onte **C**arlo) logic regression, Kooperberg and Ruczinski (2005) run an MCMC (**M**arkov **C**hain **M**onte **C**arlo) search on the whole training set, and employ the models visited after the burn-in to identify interactions that frequently occur jointly in these models. In Section 8.5, we compare logicFS with this method, and show the advantages of our approach over MC logic regression. As Section 8.3 and 8.4, this section is a modified version of Schwender and Ickstadt (2007).

Finally, we describe in Section 8.6 how the most important interactions can be selected. Since this is work in progress, only first ideas are presented.

This chapter, however, starts with the introduction of a new procedure required by logicFS for converting a logic expression into a disjunctive normal form, i.e. an OR-combination (disjunction) of AND-combinations (conjunctions).

8.2 Detecting all Prime Implicants of a Logic Expression

Let's assume that the logic expression $L = X_1 \wedge X_2^C \vee (X_3 \vee X_4) \wedge X_5^C$ is part of a logic regression model. Even though this logic expression is relatively easy to interpret, a representation of L that reveals the interacting variables directly would be preferable. Moreover, it becomes more complicated to interpret such a logic expression, the more variables it contains. Therefore, we propose to convert each logic expression into a disjunctive normal form (DNF). For the above example, the DNF is given by

$$L = \left(X_1 \wedge X_2^C\right) \vee \left(X_3 \wedge X_5^C\right) \vee \left(X_4 \wedge X_5^C\right).$$

The advantage of this representation of L is that the interactions are directly identifiable, since they are given by the conjunctions $X_1 \wedge X_2^C$, $X_3 \wedge X_5^C$, and $X_4 \wedge X_5^C$. If at least one of these conjunctions is true, then L will also be true.

To avoid redundancy, the DNF should only consist of prime implicants, i.e. minimal conjunctions. If, e.g., both $X_1 \wedge X_2 \wedge X_3$ and $X_1 \wedge X_2 \wedge X_3^C$ are part of a logic expression L, then X_3 will be redundant, since the outcome of L does not depend on the outcome of X_3. Thus, the two conjunctions can be combined to the prime implicant $X_1 \wedge X_2$.

The classical way to convert a logic expression into a (minimum) disjunctive normal form is the Quine-McCluskey algorithm (Quine, 1952; McCluskey,

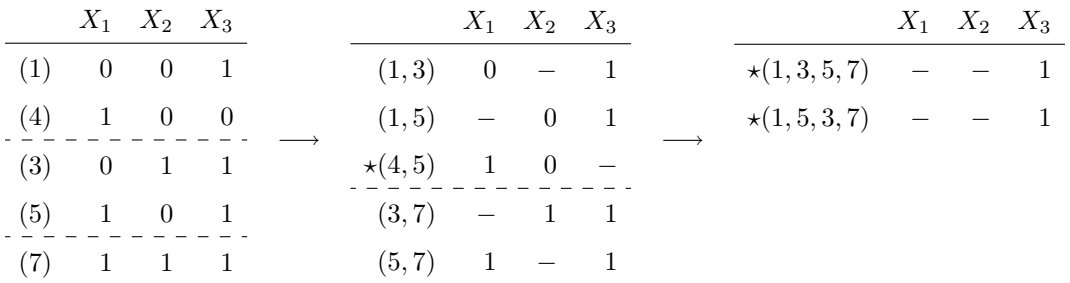

FIGURE 8.2. Quine-McCluskey algorithm. The identified prime implicants $X_1 \wedge X_2^C$ and X_3 are marked by a star. The numbers in the brackets are the decimal numbers corresponding to the binary numbers composed by the entries of the (combined) rows.

1956) consisting of two steps: Firstly, all prime implicants belonging to a logic expression are identified. Secondly, the set of prime implicants is minimized.

Since we are interested in all prime implicants, we only consider the first step in which the minterms for which the logic expression of interest is true are recursively combined, where a minterm is one of the 2^m conjunctions of length/order m composed of the values of the m binary variables comprised by the logic expression. In Figure 8.2, an example for this procedure is given. The Quine-McCluskey algorithm starts with the rightmost table called $\mathbf{T}$ in Algorithm 8.1 that contains all minterms for which the logic expression of interest is true. For example, the first row of this table represents the minterm $X_1^C \wedge X_2^C \wedge X_3$, and is combined with the third row, i.e. with $X_1^C \wedge X_2 \wedge X_3$, to $X_1^C \wedge X_3$ shown in the first row of the table in the middle of Figure 8.2. After obtaining all conjunctions of order $m-1 \overset{\text{here}}{=} 2$ that can be generated by merging two of the minterms in $\mathbf{T}$ that differ only in one position, it is determined which of these new conjunctions can be combined with each other, and so on. The algorithm stops at the leftmost table of Figure 8.2, since no further combinations are possible (for more details, see, e.g., Schwender, 2007).

Since implementing the Quine-McCluskey algorithm in `R` has led to an unsatisfactory run time, we have implemented our own procedure presented in Algorithm 8.1 that has been developed particularly for solving our problem, i.e.

for converting a logic expression consisting of up to 16 literals/variables into a DNF composed of all prime implicants that typically exhibit an order not larger than 4. (A detailed description of this algorithm with examples is given by Schwender, 2007.)

Algorithm 8.1 (Identification of Prime Implicants)

Let $\mathbf{T}$ be an $n_\mathrm{T} \times m$ matrix in which each row corresponds to one of the n_T minterms for which the logic expression L of interest is true, and each column corresponds to one of the m variables X_i, $i = 1, \ldots, m$, composing L.

1. Replace each zero in $\mathbf{T}$ by -1, and set $q = 1$.

2. Let $\mathbf{A}^{(q)}$ be a $2^q \binom{m}{q} \times m$ matrix comprising each of the $2^q \binom{m}{q}$ conjunctions of order q, where

$$a_{ki}^{(q)} = \begin{cases} 1, & \text{if } X_i \text{ is part of the } k^{\text{th}} \text{ conjunction} \\ -1, & \text{if } X_i^C \text{ is part of the } k^{\text{th}} \text{ conjunction} \\ 0 & \text{otherwise} \end{cases}. \tag{8.1}$$

3. Set $\mathbf{E}^{(q)} = \mathbf{A}^{(q)}\mathbf{T}'$, and compute $\mathbf{h}^{(q)} = I\Big(\mathbf{E}^{(q)} = q\Big)\mathbf{1}_{n_\mathrm{T}}$, where the output of $I\Big(\mathbf{E}^{(q)} = q\Big)$ is a $2^q \binom{m}{q} \times n_\mathrm{T}$ matrix with elements $i_{k\ell} = I\Big(e_{k\ell}^{(q)} = q\Big)$.

4. Set $\mathcal{H}_q = \Big\{k : h_k^{(q)} = 2^{m-q}\Big\}$. If $\mathcal{H}_q = \emptyset$, set q to $q + 1$, and repeat Steps 2-6. Otherwise, update $\mathbf{A}^{(q)}$ by removing all rows $k \notin \mathcal{H}_q$ from $\mathbf{A}^{(q)}$ such that $\mathbf{A}^{(q)}$ becomes a $|\,\mathcal{H}_q| \times m$ matrix.

5. Denote the set of identified prime implicants $P_1, \ldots, P_{G_q}$ each of order q or lower by $\mathcal{L}_q$, where $\mathcal{L}_0 = \emptyset$. If $\mathcal{L}_{q-1} = \emptyset$, add the conjunctions represented by the (remaining) rows of $\mathbf{A}^{(q)}$ to $\mathcal{L}_{q-1}$ to generate $\mathcal{L}_q$. Otherwise,

   (a) let $\mathbf{M}_{q-1}$ denote a $G_{q-1} \times m$ matrix in which each row represents – analogous to (8.1) – one of the G_{q-1} prime implicants of an order lower than q,

(b) set $\mathbf{v}^{(q-1)} = \text{diag}\left(\mathbf{M}_{q-1}\mathbf{M}'_{q-1}\right)$, i.e. let $\mathbf{v}^{(q-1)}$ be the vector containing the G_{q-1} diagonal elements of $\mathbf{M}_{q-1}\mathbf{M}'_{q-1}$, and $\mathbf{U}^{(q-1)} = \mathbf{M}_{q-1}\mathbf{A}^{(q)\prime}$,

(c) update $\mathbf{A}^{(q)}$ by removing any row of $\mathbf{A}^{(q)}$ corresponding to a non-zero entry in

$$\mathbf{1}'_{G_{q-1}} I\left(\mathbf{U}^{(q-1)} = \mathbf{v}^{(q-1)}\right),$$

where $I\left(\mathbf{U}^{(q-1)} = \mathbf{v}^{(q-1)}\right)$ is a $G_{q-1} \times |\mathcal{H}_q|$ matrix with elements $i_{k\ell} = I\left(u_{k\ell}^{(q-1)} = v_k^{(q-1)}\right)$,

(d) and add the conjunctions corresponding to the remaining rows of $\mathbf{A}^{(q)}$ to $\mathcal{L}_{q-1}$ to generate $\mathcal{L}_q$.

6. Stop if all elements of $\mathbf{1}'_{G_q} I\left(\mathbf{M}_q\mathbf{T}' = \mathbf{v}^{(q)}\right)$ are non-zero. Otherwise, set q to $q+1$, and repeat Steps 2-6.

This algorithm implemented in the `R` package `logicFS` (see Appendix C.2) is based on the fact that following the Quine-McCluskey algorithm we have to recursively combine $m-q$ times two rows of $\mathbf{T}$ to obtain a prime implicant of order q. Thus, 2^{m-q} of the rows of $\mathbf{T}$ must contain a 1 or a -1 in each of the columns corresponding to one of the variables or the complements of variables, respectively, composing a prime implicant of order q.

A problem of this idea is that if, e.g., X_i is a prime implicant, then not only 2^{m-1} rows of $\mathbf{T}$ will contain a 1 in the i^{th} column, but also 2^{m-2} rows will comprise both a 1 in the i^{th} column and a 1 (or a -1) in the k^{th} column, $k = 1, \ldots, m, k \neq i$, such that $X_i \wedge X_k$ $\left(\text{or } X_i \wedge X_k^C\right)$ will also be identified as prime implicant. To avoid this, such conjugations are removed in Step 5 of Algorithm 8.1.

The essential difference between Algorithm 8.1 and the Quine-McCluskey approach is that Algorithm 8.1 starts at $q = 1$ and successively increases q, whereas the Quine-McCluskey algorithm starts at $q = m$ and goes downwards. The former proceeding is an advantage if the orders of the prime implicants are

relatively small – as, e.g., in the logic regression models in which we typically observe interactions of a maximum order of 4 (or rarely 5).

8.3 Identification of Interesting Interaction

While Kooperberg and Ruczinski (2005) run MC logic regression once on the whole data set, we propose a procedure called logicFS (see Algorithm 8.2) in which the default search algorithm of logic expression, i.e. simulated annealing, is repeatedly applied to subsets of the data to identify variables and interactions associated with the response.

Algorithm 8.2 (logicFS – Identification of Interesting Interactions)
Given: Data of m binary variables for n observations, and the number B of iterations.

1. Draw a bootstrap sample of size n from the n observations.
2. Build a logic regression model based on this bootstrap sample.
3. Convert each logic expression comprised by the logic regression model into a disjunctive normal form consisting of prime implicants.
4. Repeat Steps 1-3 B times.
5. For each of the identified prime implicants, compute the value of an appropriate importance measure.

Some of the detected prime implicants are very important for prediction, whereas other interactions are not important at all, or might even be obstructive for a good prediction. It is therefore necessary to quantify the importances of the identified interactions.

A naive importance measure that can be computed in any classification or regression problem is the proportion of models containing the prime implicant of interest P_g, $g = 1, \ldots, G$, i.e.

$$\text{VIM}_{\text{Adhoc}}(P_g) = \frac{1}{B} \sum_{b=1}^{B} I\big(P_g \in \mathcal{L}_b\big), \tag{8.2}$$

where $\mathcal{L}_b$ is the set of identified prime implicants in iteration b, $b = 1, \ldots, B$.

In MC logic regression, a measure similar to (8.2) is determined to quantify the importances of variables and combinations of variables. In this approach, the models visited after the burn-in are employed to compute for each variable, each pair and each triplet of variables the proportion of models in which the respective variables appear jointly (but not necessarily combined by an $\wedge$) in the same logic tree. The combinations of variables occurring most frequently are then assumed to be the most important interactions.

However, some SNP interactions are explanatory only for a small subset of patients. Such interactions will hardly be found, and it is likely that they appear only in very few of the models. They would therefore be called unimportant by (8.2), even though they are actually very important for the correct prediction of some of the patients. Moreover, in a discrimination problem, a suitable measure should quantify how much a particular interaction improves the classification. This improvement should not be computed on the same data set on which the classification rule has been trained, but on an independent data set consisting of new observations.

Since in logicFS logic regression models are constructed based on subsets of the data, the respective oob observations can be employed to estimate the importance of each of the identified interactions.

If in logicFS the single tree approach is applied to a discrimination problem, we therefore propose to quantify the importance of a prime implicant P_g, $g = 1, \ldots, G$, by

$$\text{VIM}_{\text{Single}}(P_g) = \frac{1}{B} \left(\sum_{b:\, P_g \in \mathcal{L}_b} \left(N_b - N_b^{(-g)} \right) + \sum_{b:\, P_g \notin \mathcal{L}_b} \left(N_b^{(+g)} - N_b \right) \right), \tag{8.3}$$

where

N_b	is the number of oob observations in the b^{th} iteration that are correctly classified by the logic regression model constructed in the b^{th} iteration,
$N_b^{(-g)}/N_b^{(+g)}$	is the number of oob observations correctly classified by the b^{th} model after P_g has been removed from / added to the model.

We thus compare how well the logic regression models perform when P_g is part of the logic expressions and when it is not, to get a measurement of the influence of P_g on the correct classification.

In the multiple tree case, it is not possible to unambiguously add an interaction to one of the logic trees, since it is not clear to which of the logic expressions it should be appended. The prime implicant P_g, $g = 1, \ldots, G$, is therefore only removed from (and not added to) the models, and the multiple tree measure is given by

$$\text{VIM}_{\text{Multiple}}(P_g) = \frac{1}{B} \sum_{b:\, P_g \in \mathcal{L}_b} \left(N_b - N_b^{(-g)} \right). \tag{8.4}$$

This multiple tree measure is similar to the variable importance measure of Random Forests. The only difference is that Breiman (2001) permutes the values of the considered variable X_i once, and computes $N_b^{(-i)}$ based on the permuted values. By contrast, we remove the prime implicant P_g, and calculate $N_b^{(-g)}$ based on the model without this variable/interaction, since a prime implicant can be removed from a logic tree in disjunctive normal form without destroying the structure of the remaining tree.

For a particular interaction, a large value of both (8.3) and (8.4) corresponds to a high importance of this prime implicant, whereas a value of about zero leads to the assumption that the interaction has no importance for classification. A prime implicant showing a negative importance is obstructive for a good classification, as the number of misclassifications will increase if this interaction is added to the model.

8.4 Application to SNP Data

8.4.1 Simulated Data

To investigate if logicFS is able to identify the influential interactions in case-control studies, we employ two simulations: In the first simulation, we are particularly interested in the stability of the results of logicFS, and in a comparison of $\text{VIM}_{\text{Adhoc}}$ with $\text{VIM}_{\text{Single}}$ and $\text{VIM}_{\text{Multiple}}$. The goal of the second simulation is to determine if our procedure can cope with real association studies in which single interactions might have moderate effects and a relatively high percentage of the cases cannot be classified by the measured SNPs.

Simulation 1. To examine the former issue, we consider the data set of Simulation 1 (see Appendix A.4) in which each of the 500 cases but none of the 500 controls is explained by one of four interactions of different order (see Table 8.1). Using $B = 100$ bootstrap samples and allowing a maximum of 20 variables in each of the logic regression models, logicFS is applied to this data set twice – once with the single tree approach, and once with the multiple tree method allowing three logic trees to grow. Afterwards, the three importance measures are computed for each of the identified prime implicants in the respective approach.

TABLE 8.1. $\text{VIM}_{\text{Single}}$, $\text{VIM}_{\text{Multiple}}$, and $\text{VIM}_{\text{Adhoc}}$ – each averaged over 50 applications of logicFS to the data set of Simulation 1 – for the interactions explaining the specified number of cases.

		Single Tree		Multiple Trees	
	Cases	$\text{VIM}_{\text{Single}}$	$\text{VIM}_{\text{Adhoc}}$	$\text{VIM}_{\text{Multiple}}$	$\text{VIM}_{\text{Adhoc}}$
S_{12}	50	15.48	0.88	10.57	0.86
$S_{21}^C \wedge S_{32}$	100	25.48	0.74	20.10	0.70
$S_{41}^C \wedge S_{51}^C \wedge S_{61}^C$	200	54.51	0.34	26.56	0.41
$S_{72} \wedge S_{82}$	150	35.94	0.82	31.78	0.72

This procedure is repeated 50 times leading to the median importances of the four explanatory interactions displayed in Table 8.1.

This table reveals that $\text{VIM}_{\text{Single}}$ identifies these four interactions in the correct order of their importance for classification (and as the four most important prime implicants as Figure 8.3 shows). By contrast, in both the single and the multiple tree approach, $\text{VIM}_{\text{Adhoc}}$ also detects the single variable S_{12} and the two two-way interactions $S_{21}^C \wedge S_{32}$ and $S_{72} \wedge S_{82}$ as important prime implicants, but in a wrong order. Furthermore, the most important interaction $S_{41}^C \wedge S_{51}^C \wedge S_{61}^C$ shows only a moderate importance. $\text{VIM}_{\text{Adhoc}}$ is thus affected by the number of variables composing the interaction. $\text{VIM}_{\text{Multiple}}$ is also able to identify the explanatory interactions in almost the correct order. Only the importance of $S_{41}^C \wedge S_{51}^C \wedge S_{61}^C$ is underestimated which is partly due to the dependence of $\text{VIM}_{\text{Multiple}}$ from the fraction of models containing the considered prime implicant. Another reason is shown in the right panel of Figure 8.3. Not only $S_{41}^C \wedge S_{51}^C \wedge S_{61}^C$ shows up as important, but also the three two-way interactions composed of S_{41}^C, S_{51}^C and S_{61}^C which reduces $\text{VIM}_{\text{Multiple}}\big(S_{41}^C \wedge S_{51}^C \wedge S_{61}^C\big)$.

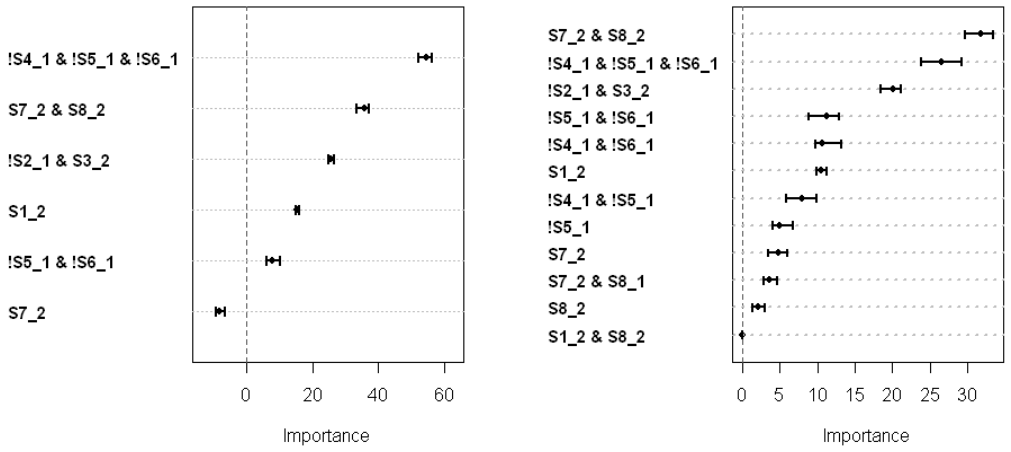

FIGURE 8.3. $\text{VIM}_{\text{Single}}$ (left panel) and $\text{VIM}_{\text{Multiple}}$ (right panel) for the prime implicants identified in all 50 iterations of the application of logicFS to the data set of Simulation 1. For each of the six or twelve interactions, the solid dot represents the median, and the bold line the IQR of the 50 values of $\text{VIM}_{\text{Single}}$ or $\text{VIM}_{\text{Multiple}}$, respectively. "!" denotes the complement of a variable, and "&" is synonymous to $\wedge$.

In Figure 8.3, only the importances of the prime implicants detected in all 50 applications of logicFS are displayed. None of the other 528 (single tree) or 7,657 (multiple tree) prime implicants found in at least one but not all of the 50 repeats shows a value of $\text{VIM}_{\text{Single}}$ or $\text{VIM}_{\text{Multiple}}$ that is larger than the importances of the four explanatory interactions in the respective iteration.

Figure 8.3 also reveals that $\text{VIM}_{\text{Single}}$ provides stable estimates for the importances of the interactions, whereas in the multiple tree case the variation in $\text{VIM}_{\text{Multiple}}$ increases with the number of variables composing the prime implicant. This is, again, mostly due to its dependence from the fraction of models containing this interaction.

Simulation 2. As a second simulation, the SNP data of Simulation 2 are considered that are more realistic for a genetic association study. This simulation – explained in more details in Appendix A.4 – consists of 50 data sets each containing data of 1,000 observations and 50 SNPs. The case-control status of each observation is specified by a random draw from a Bernoulli distribution, where the probability for being a case depends on the presence of the two logic expressions $S_{61} \wedge S_{71}^C$ and $S_{31}^C \wedge S_{91}^C \wedge S_{10,1}^C$. This probability is 0.378 even if an observation exhibits none of these interactions intended to be influential for the risk of developing the disease of interest. A reason for this might be that there are other genetic (or environmental) factors that have not been surveyed in this association study, but also have an impact on the disease risk.

Both the single tree approach with a maximum of six variables and the multiple tree method with two trees and a maximum of eight variables are applied to each of these 50 data sets using $B = 50$ iterations.

Table 8.2 reveals that both $S_{61} \wedge S_{71}^C$ and $S_{31}^C \wedge S_{91}^C \wedge S_{10,1}^C$ are detected in all 50 data sets. Moreover, they are identified as the two most important logic expressions in almost any of these data sets, where $S_{61} \wedge S_{71}^C$ mostly ranks first with a mean importance of 18.88 in the single and 15.19 in the multiple tree approach, and $S_{31}^C \wedge S_{91}^C \wedge S_{10,1}^C$ ranks second with a mean importance of 12.21

TABLE 8.2. Ranks of the two SNP interactions intended to be influential for the case-control status in the applications of both the single and the multiple tree approach to each of the 50 data sets from Simulation 2.

	$S_{61} \wedge S_{71}^C$		$S_{31}^C \wedge S_{91}^C \wedge S_{10,1}^C$	
Rank	Single	Multiple	Single	Multiple
1	45	42	5	6
2	4	6	42	32
3	1	2	3	10
4	0	0	0	2

and 6.44, respectively. If one of these interactions ranks third (or lower), then the logic expressions identified to be more important will typically contain this or the other influential interaction plus another variable.

8.4.2 GENICA and HapMap Data

Using $B = 200$ iterations, logicFS is applied to the GENICA data set twice – once growing a single tree with a maximum of ten leaves, and once allowing three trees to grow with a maximum of 16 variables in all three trees combined.

In the single tree case, this leads to the detection of 1,052 potentially interesting SNPs and SNP interactions, whereas in the multiple tree application, 1,589 SNPs and SNP interactions are identified. However, as shown in Figure 8.4, just one interaction, namely $!X18$ & $X20$, or decoded

$$\text{ERCC2\_6540}_1^C \wedge \text{ERCC2\_18880}_1,$$

consisting of the two SNPs from the ERCC2 gene also identified in Section 6.3.4 and 6.4.3 seems to be associated with the case-control status. If thus ERCC2_6540 is of the homozygous reference genotype, and ERCC2_18880 is not of this genotype, then a women will have a little higher risk of developing breast cancer.

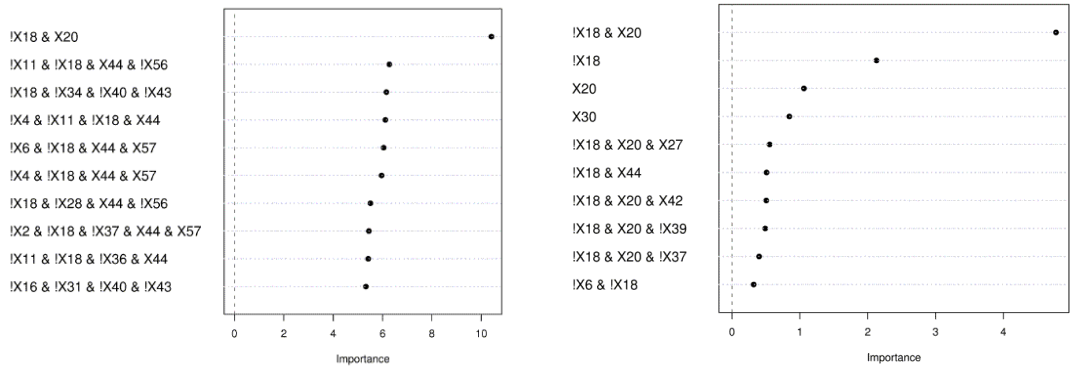

FIGURE 8.4. $\text{VIM}_{\text{Single}}$ (left panel) and $\text{VIM}_{\text{Multiple}}$ (right panel) of the most important interactions detected in the analysis of the GENICA data set. Since the SNP names are too long for graphical representation, they are coded.

If the application of logicFS is repeated, $\text{ERCC2\_6540}_1^C \wedge \text{ERCC2\_18880}_1$ will still be found as the most important interaction. But all the other interactions will typically be replaced by other prime implicants that typically contain ERCC2_6540_1^C. This and $\text{VIM}_{\text{Multiple}}\big(\text{ERCC2\_6540}_1^C\big)$ displayed in the right panel of Figure 8.4 indicate that ERCC2_6540 itself has a slight effect on the breast cancer risk (cf. also Section 6.3.4 and 6.4.3).

Using several parameter settings, we have applied logicFS several times to the reduced HapMap data set consisting of the 157 SNPs preselected by SAM (see Section 6.3.4), but have not found consistent results in these applications. While in the single tree approach at least the first 100 interactions typically show importances between 1.4 and 2.6, none of the prime implicants identified in the multiple tree approach exhibits an importance larger than 0.12. A reason for this is shown in the left panel of Figure 8.5: At least 68 binary variables – each belonging to a different SNP – are required to classify all 45 Han Chinese and all 45 Japanese correctly. Thus, not a few interactions, but a large number of SNPs are needed for solving this classification problem.

In such a case, an approach in which the importances of single variables is quantified might be better suited than logicFS. However, the application of the importance measure of Random Forests to the HapMap data also does not lead

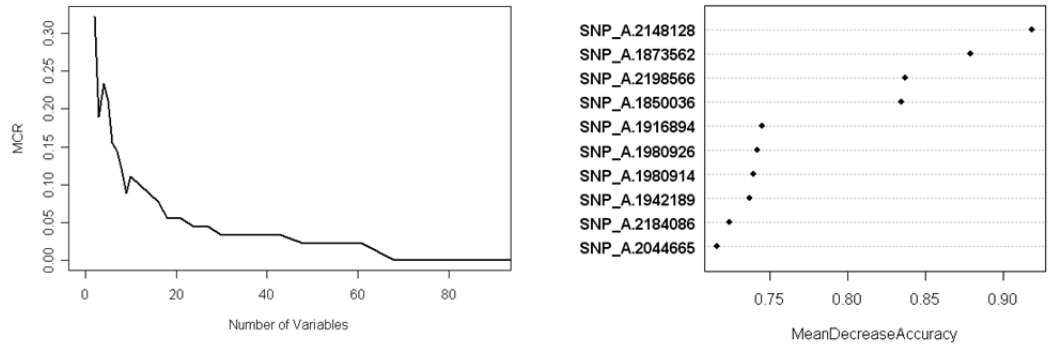

FIGURE 8.5. Left panel: Application of RFE-SVM to the reduced HapMap data set, where in each step of RFE-SVM the 10% least important variables are shaved off. Right panel: The ten most important SNPs found in an application of Random Forests to the HapMap data set using 5,000 trees and 12 randomly selected variables at each node.

to very consistent results. Only the first two SNPs shown in the right panel of Figure 8.5 are constantly detected under the top 15 SNPs (but only seldom as the two most important variables).

8.5 Comparison with MC Logic Regression

To compare logicFS with MC logic regression, the latter is applied to the simulated data sets considered in Section 8.4.1 using the same parameter settings of logic regression as in Section 8.4.1 and 500,000 iterations in the MCMC algorithm. The last 400,000 models are kept in memory to compute the importance measures. For each variable, each pair and triplet of variables, the output of MC logic regression provides the number of models containing this variable or set of variables as measure of importance, where it is ignored, on the one hand, whether the variable itself or its complement is in the model, and on the other hand, whether the variables are combined by $\wedge$ or by $\vee$. Since specific conjunctions are not considered by this importance measure, we additionally compute the value of $\text{VIM}_{\text{Adhoc}}$ for each of the prime implicants obtained by converting each of the logic trees comprised by the visited models into a DNF using Algo-

rithm 8.1. (We, however, do not calculate $\text{VIM}_{\text{Single}}$ and $\text{VIM}_{\text{Multiple}}$, since, on the one hand, all models are built during the same run of MC logic regression on the whole data set – and not in different applications to different subsets of this data set – and on the other hand, the determination of these measures would be done on the same test data for each of the iterations.)

A drawback of the measure used in MC logic regression is that interactions of different orders have to be considered separately, since each subset of the variables contained in the set of interest is in at least as many models as the set itself such that each subset is at least as important as the set of interest. By contrast, both $\text{VIM}_{\text{Single}}$ and $\text{VIM}_{\text{Multiple}}$ enable the comparison of interactions of different orders.

In Figure 8.6, the results of ten applications of MC logic regression to the data set of Simulation 1 are displayed. This figure reveals two problems of this procedure: If in the single tree case the set of interacting SNPs is detected, then it will typically be in virtually any of the models, and in almost any case, as the

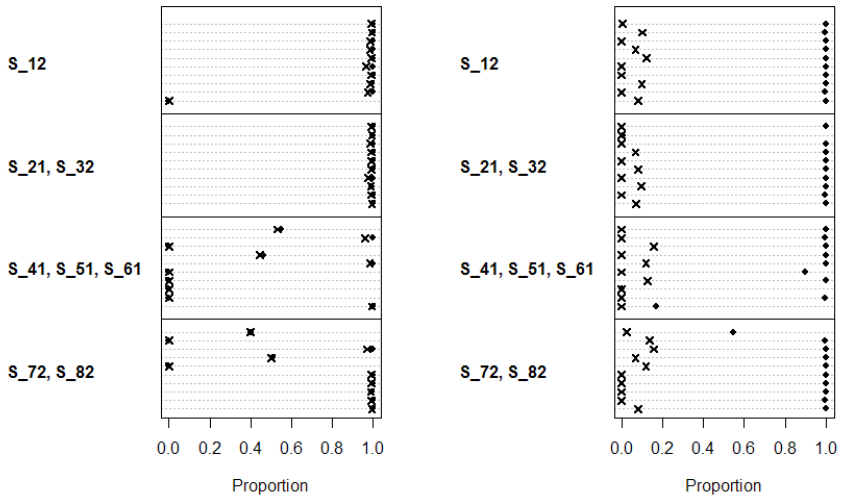

FIGURE 8.6. Fraction of models (marked by solid dots) containing particular sets of variables, and $\text{VIM}_{\text{Adhoc}}$ (marked by crosses) for specific interactions computed from the models visited during ten applications of both the single (left panel) and the multiple (right panel) tree approach of MC logic regression to the data set of Simulation 1.

intended interaction. However, even the single variable S_{12} is not found in any of the applications, and the triplet $\{S_{41}, S_{51}, S_{61}\}$ is only identified in 50% of the analyses. Even though in the multiple tree case the sets of interacting SNPs are found in almost any of the applications, the SNP interactions explaining the cases are rarely detected. For example, S_{12} is in virtually any of the models, but mostly in interaction with another variable. Moreover, not only $\{S_{21}, S_{32}\}$ and $\{S_{72}, S_{82}\}$, but also, e.g., $\{S_{32}, S_{72}\}$ or $\{S_{21}, S_{82}\}$ frequently appear jointly in more than 99% of the models, and would therefore be considered to be of a similar importance when using the proportion as importance measure. By contrast, none of the two-way interactions composed of either the pair $\{S_{32}, S_{72}\}$ or of $\{S_{21}, S_{82}\}$ exhibits a large value of $\text{VIM}_{\text{Single}}$ or $\text{VIM}_{\text{Multiple}}$ (cf. Section 8.4.1, in particular Figure 8.3).

The applications of MC logic regression to the 50 data sets of Simulation 2 lead to similar results: $S_{61} \wedge S_{71}^C$ is always identified by the single tree approach, but in only about 40% of the applications of the multiple tree approach, whereas $S_{31}^C \wedge S_{91}^C \wedge S_{10,1}^C$ is found in 90% of the single tree applications, and in about 60% of the analyses with multiple trees. By contrast, logicFS always identifies both $S_{61} \wedge S_{71}^C$ and $S_{31}^C \wedge S_{91}^C \wedge S_{10,1}^C$. (These results differ a little from the results presented in Schwender and Ickstadt, 2007. A reason for this might be that in Schwender and Ickstadt, 2007, we have split each of the 50 data sets into 63.2% training and 36.8% test data such that the distribution of the explained cases and controls remains unchanged in the split data sets. Because of this disagreement, we have repeated the above analysis a few times, where each of these analyses has led to similar results.)

These two simulations show the advantage of logicFS over MC logic regression: In MC logic regression, SNP interactions are identified by applying a search algorithm once to the whole data set. If an interaction explanatory for the case-control status is detected once, i.e. is in one of the models visited during this search, then it is very likely that it will be identified to be important. However, if the variables composing this interaction do not jointly occur in any of these

models, or are in the model but in conjunction with other variables, the actual explanatory interaction will not be detected.

By contrast, in logicFS a search algorithm is applied several times to different subsets of the data such that an interesting interaction is not lost even if it is not identified in some of the runs. Hence, logicFS stabilizes the search for interesting variables and interactions.

8.6 Selecting SNP Interactions

Currently, we follow an approach similar to the one proposed by Breiman (2001) for Random Forests: logicFS is applied to the SNP data to identify the most important interactions by taking a look on $\text{VIM}_{\text{Single}}$ and $\text{VIM}_{\text{Multiple}}$. Afterwards, these selected interactions are used in logic regression, in logistic regression, or in any other discrimination method as binary variables to construct a classification rule.

Instead of subjectively deciding which prime implicants are important, a more objective criterion might be preferred. An example for such an importance measure is a statistic for testing if the importance of a prime implicant is larger than some prespecified value μ_0, where $\mu_0 \geq 0$ can be chosen based on, e.g., biological knowledge, or the number of observations that should be explained by an interaction to be considered as important. In the single tree case, we thus compute the t-statistic

$$t_g = \sqrt{B}\,\frac{\text{VIM}_{\text{Single}}\big(P_g\big) - \mu_0}{s_g} \tag{8.5}$$

for each prime implicant P_g, $g = 1, \ldots, G$, where s_g is the sample standard deviation of the improvements

$$\text{Imp}_b\big(P_g\big) = \begin{cases} N_b - N_b^{(-g)}, & \text{if } P_g \in \mathcal{L}_b \\ N_b^{(+g)} - N_b, & \text{if } P_g \notin \mathcal{L}_b \end{cases}$$

in the b^{th} logic regression model, $b = 1, \ldots, B$, due to prime implicant P_g (cf. Section 8.3). For the multiple tree case, the t-score can be defined analogously.

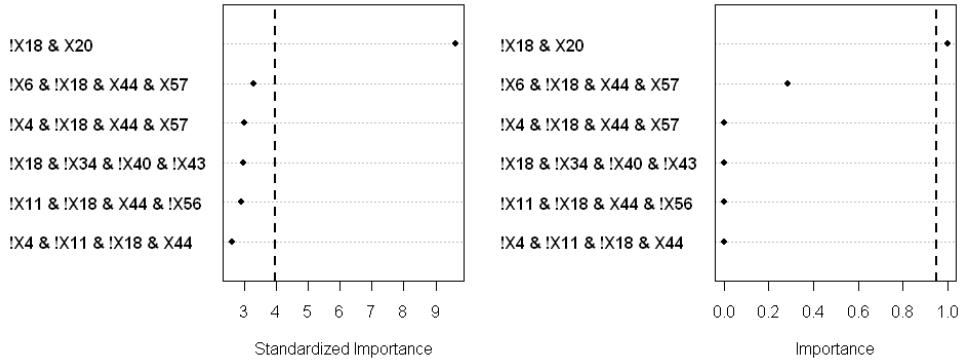

FIGURE 8.7. VIM_{Norm} (left panel) and VIM_{Perm} (right panel) for the six most important prime implicants detected in the application of the single tree approach to the GENICA data set using $\mu_0 = 5$. The dashed vertical lines are at the $(1 - 0.05/1052)$-quantile of the t_{199}-distribution, i.e. at 3.98, and at 0.95, respectively.

If t_g is larger than a prespecified threshold, e.g., the $\big(1 - 0.05/G\big)$-quantile of the t_{B-1}-distribution, then P_g will be called important. Instead of the p-value, we employ the t-statistic as importance measure – and call it VIM_{Norm} – since the selection criterion should be "The larger, the more important."

To avoid the independence and the normality assumption, a permutation method might be employed to test the importances of the prime implicants. A simple procedure would be to randomly permute the sign of $\text{Imp}_b - \mu_0$, $b = 1, \ldots, B$, i.e. to use either $\text{Imp}_b - \mu_0$, or $\mu_0 - \text{Imp}_b$, and to compute the permuted t-statistics t_{ga}, $a = 1, \ldots, A$, where A is the number of permutations. In this case, an appropriate measure of importance for P_g, $g = 1, \ldots, G$, is

$$\text{VIM}_{\text{Perm}}(P_g) = 1 - \min\left\{1, \frac{G}{A}\sum_{a=1}^{A} I\big(t_g \leq t_{ga}\big)\right\}. \tag{8.6}$$

In Figure 8.7, the results of a first applications of both (8.5) and (8.6) using $A = 50,000$ permutations to the GENICA data set are displayed. As expected, only the interaction $\text{ERCC2\_6540}_1^C \wedge \text{ERCC2\_18880}_1$ will be selected if these measures are employed for choosing interactions.

PART IV

Discussion and Additional Information

Chapter 9

Summary and Discussion

Preprocessing is an important step in the analysis of Affymetrix microarrays, since the results of this low level analysis can influence the results of high level analyses such as variable selection and classification substantially. This step is, however, often ignored, since the Affymetrix software also provides estimates for the expression values of the genes. Using these signals might not be the best idea, as our comparison of the preprocessing algorithms of Affymetrix, i.e. MAS 5.0 and PLIER, with, on the one hand, Roche in-house modifications of the latter approach, and on the other hand, the most popular academic alternative to these algorithms, i.e. RMA, and modifications of RMA reveals.

In this comparison, PLM that differs from RMA only in the summarization step shows overall the best performance. Fitting probe level models (PLMs) particularly leads to the identification of the most differentially expressed genes, and to the most parsimonious PCA representation (for a more detailed summary of the comparison, see the conclusions in Section 4.7).

Contrary to other comparisons (e.g., Wu et al., 2004), GCRMA (and GC-PLM) in which the convolution model based background correction of RMA is replaced by an approach that takes the base composition of the probe sequences into account does not improve the estimation of the gene expression values. This might be due to the use of another type of microarrays. For the `R` function of GCRMA, the base effects have been estimated by employing the Affymetrix

HG-U133A chips that are also considered in Wu et al. (2004). In our comparison, however, we have analyzed data from Affymetrix HG-U133_Plus_2 arrays. Using estimates for the base effects that are more appropriate for this type of chip and an empirical Bayes approach also proposed by Wu et al. (2004) as an alternative to the maximum likelihood estimate employed in the default version of GCRMA might improve the performance of GCRMA (and GCPLM).

Having identified PLM as the best preprocessing method in our comparison does not mean that PLM is the ultimative solution to this problem. For example, Hekstra et al. (2003) show that there is a saturation effect at high mRNA levels that exactly follows an adsorption model. Since the assumption behind all the considered preprocessing procedures that there is a linear relationship between the measured intensities and the mRNA concentrations is hence inaccurate, a preprocessing method accounting for this saturation effect might improve the detection of genuinely affected genes.

However, even the best preprocessing procedure cannot lead to good estimates of the expression values if the annotations of the genes, i.e. the information on which probe sequences belongs to which gene, is wrong. It is a well-known problem of the Affymetrix cdf environment, i.e. the file containing information on which cells on the chip comprise the probes that represent a specific gene, is based on outdated knowledge. A solution to this problem is to replace this file by an alternative cdf environment that takes the latest knowledge on the sequences of the genes available at public repositories such as RefSeq into account (cf. Dai et al., 2005).

Another problem – in particular, when using `R` – is the high-dimensionality of the data. For each Affymetrix HG-U133_Plus_2 chip, e.g., the intensities of more than 600,000 probe pairs have to be reduced to 54,675 expression values – one value for each probe set on this chip. If multi-chip methods such as PLM or PLIER are applied to studies comprising several tens of chips using the standard `R` functions and a computer with a moderate amount of RAM, this computation will fail because of massive memory problems.

In this thesis, we have proposed a strategy based on repeatedly in- and outputting the probe intensities that makes it possible to apply PLM to several hundred Affymetrix HG-U133_Plus_2 chips. This approach can easily be modified for the other multi-chip methods also considered in this thesis. Furthermore, this procedure is not restricted to DNA microarrays, but can also be used to preprocess Affymetrix chips measuring the genotypes of hundreds of thousands of SNPs. In this application, however, not only the probe sets have to be split into several chunks, but also the data from one chip should be stored in different files.

Having determined the expression values of the genes or the genotypes of the SNPs, variable selection and classification methods can be applied to these data. Since many approaches for such high level analyses of gene expression data have been proposed in recent years, and the goals in these analyses are similar to the aims in studies concerned with genotype data, we have shown how methods that can cope with the high-dimensionality of DNA microarrays can be adapted to SNP data. These modified procedures can then not only be applied to SNP microarray data, but also to the genotypes of the (few ten) SNPs considered in an association study.

As a first example, a procedure called kNNimpute for imputing missing values based on k nearest neighbors has been adapted to SNP data and compared with other imputation methods in their application to the GENICA data set. Since KNNimpute leads to the lowest fraction of falsely imputed values, the missing values of the GENICA data set have been replaced by this approach.

Afterwards, the Significance Analysis of Microarrays (SAM) has been adapted to categorical data by replacing a moderated t-statistic and its null distribution by Pearson's χ^2-statistic and its null distribution. This method has then been applied to the HapMap data to reduce the number of SNPs from 121,774 to 157, where this subset of SNPs exhibits an estimated FDR of 6.94%. In the analysis of the GENICA data set, SAM reveals that the SNP ERCC2_6540 (ref SNP ID: rs1799793) has a slight effect on the breast cancer risk which is increased

if this SNP is considered in interaction with ERCC2_18880 (rs1052559). This result supports the findings of Justenhoven et al. (2004).

As a third example, we have adopted the ideas of the Prediction Analysis of Microarrays (PAM) to develop a discrimination method for SNP data, and have applied this procedure to both the GENICA and the HapMap data. The results of the analysis of the former data set are similar to the results of SAM: If individual SNPs are considered, only ERCC2_6540 will be required for classification. If two-way interactions are taken into account, then the interaction of the two SNPs from the gene ERCC2 will form the classification rule. The application of PAM to the HapMap data shows that it can be an advantage to reduce the set of SNPs first, and then construct the classification rule based on the resulting subset, as the classifier based on the 157 SNPs preselected by SAM decreases the misclassification rate of the rule built by considering all 121,774 SNPs substantially.

A drawback of the Prediction Analysis of Categorical Data is the class prediction. In the original version of PAM, the class of a new observation is predicted by the group showing the smallest distance between its (shrunken) centroid and the expression values of this observation. Since in genotype data such a distance cannot be computed, and in PAM the genes are considered individually, the Prediction Analysis of Categorical Data employs a naive Bayes classifier. Reducing the set of SNPs by shrinking the group-wise test statistics successively towards zero, and predicting the class of a new observation by, e.g., a kNN classifier using this subset of SNPs might improve the class prediction.

A problem of all three adapted procedures is that tens or even hundreds of thousands of statistics have to be determined in each of the applications. Since calculating each of these scores individually can be very time-consuming in `R`, we have shown how matrix algebra can be employed to compute all the statistics simultaneously which reduces the computation time substantially.

Only a small number of variable selection and discrimination methods have been developed particularly for the analysis of genotype data. One of the few

exceptions is logic regression that has shown a good performance in comparisons with other discrimination and regression procedures in their application to SNP data. To investigate if this is also true for our real and simulated data, we have applied logic regression and other discrimination methods such as SVM and Random Forests to these data sets.

The results of this comparison show that logic regression will lead to a lower misclassification rate than other procedures, if a few interactions are explanatory for the case-control status. If, however, many SNPs are required to construct a good classification rule, then other discrimination methods outperform logic regression. In such a case, applying bagging to logic regression can stabilize this approach, and thus, reduce its misclassification rate.

However, not only bagging, but also other ensemble methods such as boosting (e.g., Freund and Shapire, 1996) can be employed to stabilize logic regression. Furthermore, it would be interesting to investigate, if – contrary to bagging – boosting is able to reduce the misclassification rate of logic regression in the analysis of the GENICA data set, or of similar data sets from other association studies.

As in the other analyses, the comparison of the discrimination methods shows that the only two SNPs of the GENICA study that might have an effect on the breast cancer risk are ERCC2_6540 and ERCC2_18880, since the best classifier would be to predict the class of a new observation based on the joint distribution of these two SNPs.

Logic regression cannot only be used for classification, but also for the identification of interesting interactions of binary variables. This is particularly important in the analysis of SNP data, since not individual SNPs, but high-order interactions of SNPs are assumed to be responsible for a higher risk of developing a complex disease such as cancer.

In this thesis, we have introduced, on the one hand, a procedure called logicFS that enables the detection of such interactions using a bagging version of logic regression, and on the other hand, two measures, $\text{VIM}_{\text{Single}}$ and

$\text{VIM}_{\text{Multiple}}$, for quantifying the importance of each of the identified interactions for classification in case-control studies.

In the applications of logicFS to the simulated SNP data sets, all logic expressions / interactions intended to be explanatory for the case-control status of the observations are always found, where they show the largest values of $\text{VIM}_{\text{Single}}$ and $\text{VIM}_{\text{Multiple}}$ in virtually any of these applications.

In the analysis of the GENICA data set, $\text{ERCC2\_6540}_1^C \wedge \text{ERCC2\_18880}_1$ is detected as the only interaction that has a slight influence on the breast cancer risk. If thus ERCC2_6540 is of the homozygous reference genotype, and ERCC2_18880 is not of this genotype, then a women will have a little higher risk of developing breast cancer.

The applications of logicFS to the HapMap data do not lead to consistent results. This might be due to the large number of SNPs required for the distinction between the Han Chinese and the Japanese, and shows a limitation of logicFS: If a few interactions are explanatory for the covariate of interest – which is the data situation for which logicFS has been developed – then logicFS is well-suited for the analysis of this data. If, however, a large number of single variables are needed for the construction of a good classification rule, then other methods might be more appropriate.

Advantages of logicFS over other approaches such as the importance measure of Random Forests are, on the one hand, that it allows to compute the importances of interactions without using these interactions as input variables, and on the other hand, that the genotypes responsible for the higher disease risk are revealed directly by the prime implicants identified by logicFS.

An advantage of logicFS over MC logic regression (Kooperberg and Ruczinski, 2005) is that the search for explanatory interactions is stabilized by running a search algorithm several times on several subsets of the data. Contrary to using the fraction of models containing a particular set of variables as importance measure, $\text{VIM}_{\text{Single}}$ and $\text{VIM}_{\text{Multiple}}$ can, on the one hand, quantify the importance of a specific interaction / prime implicant for classification – and

not just for a set of variables – and on the other hand, be employed for the comparison of interactions of different orders. The latter is not directly possible when considering fractions, as each subset of variables contained in the set of interest will show at least the same importance as this set of variables.

A major goal in case-control studies is the construction of a classification rule based on as few variables as possible. Having identified variables and interactions of variables, and quantified their importance, the most important features can be selected for building such a rule. This set of variables/interactions might either be chosen subjectively by taking a look on the importances, or more objectively by, e.g., testing these importances. In this thesis, we have presented first ideas based on a parametric and a permutation based t-test for selecting the most important interactions, and applied these selection approaches to the GENICA data set leading to the identification of $\text{ERCC2\_6540}_1^C \wedge \text{ERCC2\_18880}_1$ as the only important interaction. This, however, is work in progress, and we will take a closer look on these and other procedures in the near future.

Since we are mainly interested in case-control studies, we have only proposed variable importance measures for the analysis of data with a binary response. However, logicFS is not restricted to this type of analysis. It can also be applied to other types of studies such as QTL (**Q**uantitative **T**rait **L**oci) studies in which SNPs are employed as predictors for quantitative responses. In this case, an appropriate importance measure might, e.g., be based on the sums of squares that would replace the numbers of correctly classified observations in (8.3) and (8.4). This, however, would make it additionally necessary to change the signs of the differences in (8.3) and (8.4).

Moreover, logic regression and thus logicFS are not restricted to the use of simulated annealing as search algorithm. Other approaches such as genetic programming or a greedy search can also be employed in the search for the best logic regression model. Using, e.g., a greedy search would reduce the run time of logicFS considerably.

Logic regression can handle quantitative responses. However, continuous

explanatory variables cannot be included in the logic trees, unless they are dichotomized before they are used in logic regression. Approaches how such a dichotomization can be done are considered by Schmitt (2005). It, however, might be a better idea to split the variables within the algorithm – as, e.g., in CART – since splits that work best on the whole data set might be suboptimal dichotomizations when subsets of these data are considered. We therefore plan to extend logic regression to enable the inclusion of categorical and continuous variables into the logic trees such that, e.g., interactions between SNPs and continuous environmental variables can be detected. A first step in this direction has already been done: The genetic programming based logic regression introduced by Ickstadt et al. (2006a) allows the inclusion of the SNPs themselves by employing multi-valued logic such that the splitting of each SNP into two dummy variables is not necessary anymore.

All approaches introduced in this thesis are implemented in `R` packages (see Appendix C). While the `R` packages `siggenes` and `logicFS` containing functions for the SAM analysis of categorical data and for logicFS, respectively, can be downloaded from the web page of the BioConductor project, all other methods are available in the `R` package `c7Tools` that will also be published at http://www.bioconductor.org in the near future, and will be extended by functions for other procedures for the analysis of genotype data developed at the Collaborative Research Center 475 of the University of Dortmund.

Appendix A

Data Sets

A.1 Affymetrix HG-U133_Plus_2 Chips

(*This section is a modified excerpt from Schwender and Belousov, 2006.*)

The 38 Affymetrix HG-U133_Plus_2 chips used in the comparison of preprocessing procedures presented in Chapter 4 are originally from two projects – a breast (BRCA; 20 arrays) and a colorectal cancer (CRCA; 18 arrays) project.

The measurements of the human tissue material have been performed according to the standard Affymetrix workflow. Replicate measurements of each tissue sample have been done to access the technical (workflow) variability. At each of the steps of the Affymetrix workflow (see Table A.1), complete randomization has been performed to avoid possible confounding and biases.

TABLE A.1. Steps of the Affymetrix workflow, and the different protocols/specifications used at each of these steps.

Project	RNA Extraction Protocol	Amplification Protocol	total-RNA $[\mu g]$	IVT time $[h]$	Labeling Protocol
CRCA	Qiagen	Affy 1-cycle	$1, 5, 10$	$4, 16$	ENZO
BRCA	Qiagen, Qiagen LT, TriPure	Affy 1-cycle	$2, 5, 10$	16	Invitrogen

A.2 GENICA

The GENICA study is carried out by the Interdisciplinary Study Group on **G**ene **EN**vironment **I**nteraction and Breast **CA**ncer in Germany, a joint initiative of researchers dedicated to the identification of genetic and environmental factors associated with sporadic breast cancer. This age-matched and population-based case-control study has been initially launched within the activities of the German Human Genome Project (DHGP, http://www.dhgp.de), and continues until present (for details, see http://www.genica.de).

Even though exogenous factors such as reproduction variables, hormone variables, and life style factors have also been assessed, the focus in this thesis is on a subset of the genotype data from the GENICA study. More precisely, data of 1,234 women (609 cases and 625 controls) and 39 SNPs belonging to the estrogen, the DNA repair, or the control of cell cycle pathway are available for the analyses.

Since a few of the women show a large number of missing genotypes, all observations with more than three missing values are removed from the analysis leading to a total of 1,199 women (592 cases and 607 controls).

Note that the data set used in this thesis is from the latest version of the GENICA data set that will be used in upcoming publications of the GENICA project (e.g., in Justenhoven et al., 2006). In other publications, other versions of this data set have been used: In Schwender et al. (2004), a first data set comprising the genotypes of 25 SNPs is considered. Ickstadt et al. (2006b) analyze a former version of the GENICA data set consisting of 77 SNPs from which 40 are employed in Schwender and Ickstadt (2007).

A.3 HapMap

The International HapMap Project (http://www.hapmap.org; The International HapMap Consortium, 2003) is a collaboration of several scientific groups from

different countries. The goals of this project are the development of a haplotype map of the human genome and the comparison of genetic variations of individuals from different populations. To achieve this goal, millions of SNPs have been genotyped for each of 270 people from four different populations.

In this thesis, the SNP data of 45 unrelated Han Chinese from Beijing and 45 unrelated Japanese from Tokyo measured by employing the Affymetrix GeneChip Mapping 500K Array Set are considered.

This array set consists of two chips (the Nsp and the Sty array named after the restriction enzymes used on these chips) each enabling the genotyping of about 250,000 SNPs. In this thesis, we focus on the BRLMM genotypes (see Section 3.7) of the 262,264 SNPs from the Nsp array that can be downloaded from http://www.affymetrix.com/support/datasets.affx.

All SNPs showing one or more missing genotype (54,400 SNPs), for which not all three genotypes are observed (75,481 SNPs), or that have a minor allele frequency less than or equal to 0.1 (10,609 SNPs) are excluded in this order from the analysis leading to a data set composed of the genotypes of 121,774 SNPs and 90 individuals.

A.4 Simulation

Simulation 1. In the first simulation, data of 1,000 observations (500 cases and 500 controls) and 50 SNPs are simulated using the `R` function `simulateSNPs` (see Appendix C.3). An observation is classified as case, if one of the following logic expressions is true:

- $\{S_1 = 2\} = S_{12}$ (explains 50 cases),
- $\{S_2 = 0\} \wedge \{S_3 = 2\} = S_{21}^C \wedge S_{32}$ (100 cases),
- $\{S_4 = 0\} \wedge \{S_5 = 0\} \wedge \{S_6 = 0\} = S_{41}^C \wedge S_{51}^C \wedge S_{61}^C$ (200 cases),
- $\{S_7 = 2\} \wedge \{S_8 = 2\} = S_{72} \wedge S_{82}$ (150 cases),

where $S_i = 0$, if S_i is of the homozygous reference genotype, and $S_i = 2$, if it is of the homozygous variant genotype.

Apart from the SNPs explaining the cases, the genotypes are randomly drawn such that the minor allele frequency of each SNP lies between 0.2 and 0.4.

Simulation2. As a second simulation, SNP data of 1,000 observations and 50 SNPs are generated using the function `simulateSNPfblr` (see Appendix C.3), where each SNP exhibits a minor allele frequency of 0.25. The case-control status y of each observation is randomly drawn from a Bernoulli distribution with mean $\text{Prob}(Y = 1)$, where

$$\text{logit}\big(\text{Prob}(Y = 1)\big) = -0.5 + 1.5L_1 + 1.5L_2$$

with

$$L_1 = \big\{S_6 \neq 0\big\} \wedge \big\{S_7 = 0\big\} = S_{61} \wedge S_{71}^C$$

$$L_2 = \big\{S_3 = 0\big\} \wedge \big\{S_9 = 0\big\} \wedge \big\{S_{10} = 0\big\} = S_{31}^C \wedge S_{91}^C \wedge S_{10,1}^C.$$

This procedure is repeated 50 times such that 50 data sets are generated. For each of the different probabilities for being a case, the mean numbers of cases and controls over these data sets are summarized in Table A.2.

TABLE A.2. Probabilities for being a case when showing none, one, or both of the influential interactions, and the mean numbers of cases and controls over the 50 data sets of Simulation 2.

Interactions	Probability	Cases	Controls
0	0.378	232	388
1	0.731	245	91
2	0.924	40	3

Appendix B

Supplementary Material

B.1 Supplementary Plots

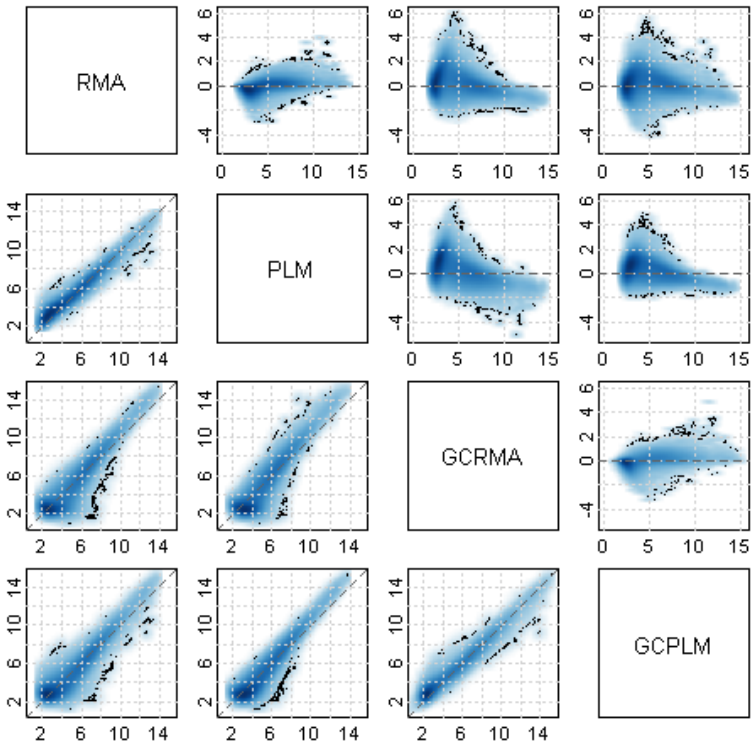

FIGURE B.1. Pairwise smoothed scatter (lower triangle) and MA (upper triangle) plots of the expression values of the four RMA approaches. The darker the color, the higher is the density at this point. The 500 signals with the lowest densities are marked by a black dot.

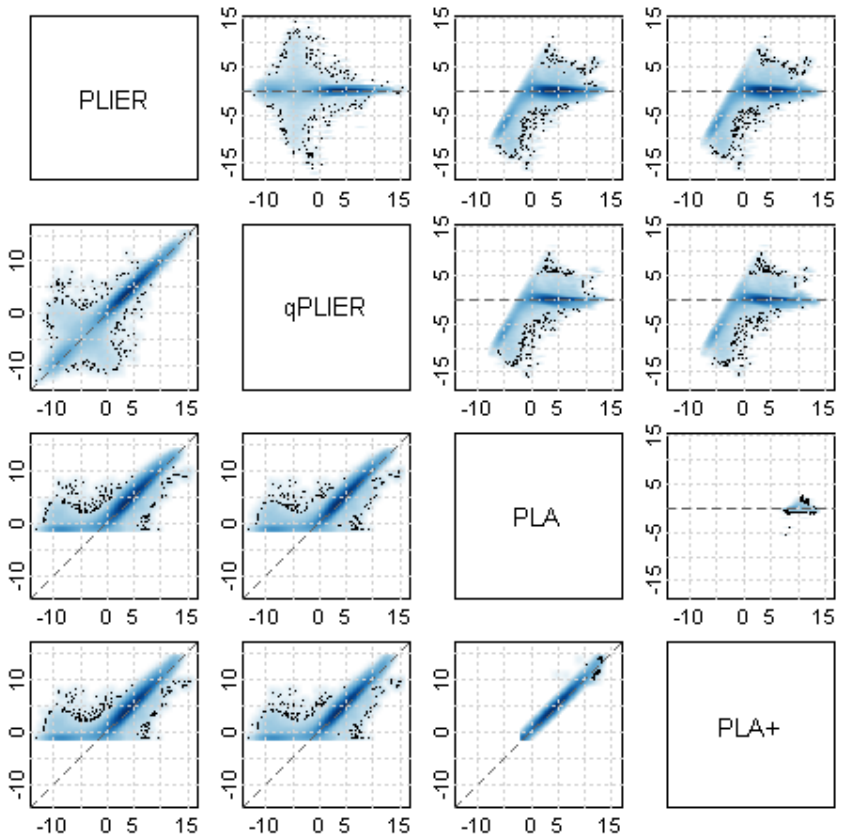

FIGURE B.2. Pairwise smoothed scatter (lower triangle) and MA (upper triangle) plots of the expression values of the PLIER and PLIER-like algorithms. The darker the color, the higher is the density at this point. The 500 signals with the lowest densities are marked by a black dot.

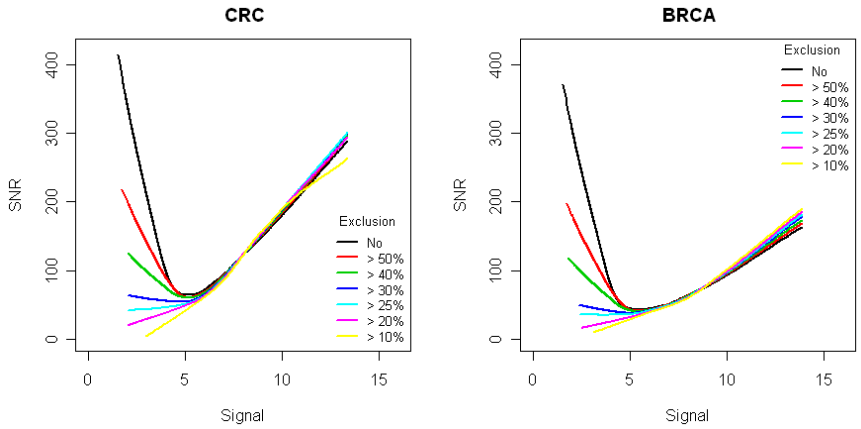

FIGURE B.3. Signal-to-Noise Ratio of GCRMA when probe sets exhibiting more than a particular percentage of truncated background corrected PM intensities are excluded from the comparison in Section 4.4.

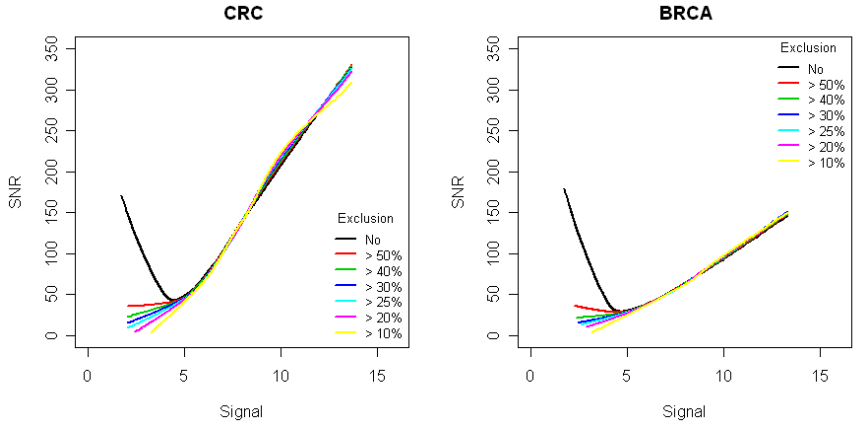

FIGURE B.4. Signal-to-Noise Ratio of GCPLM when probe sets showing more than a particular percentage of truncated background corrected PM intensities are excluded from the comparison in Section 4.4.

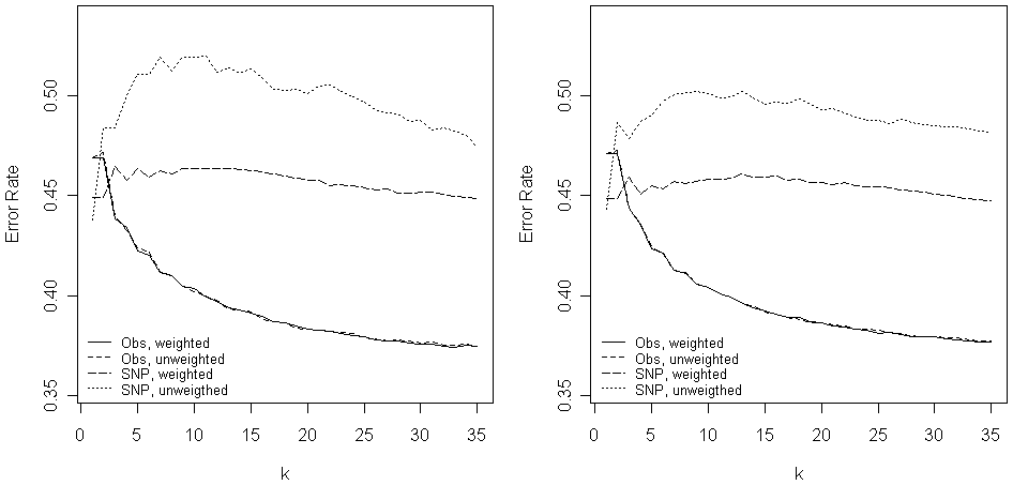

FIGURE B.5. Fractions of falsely imputed values if (weighted) k nearest neighbors is applied to either the observations or the SNPs to impute the 5% (left panel) or 10% (right panel) artificially generated missing values in the GENICA data set.

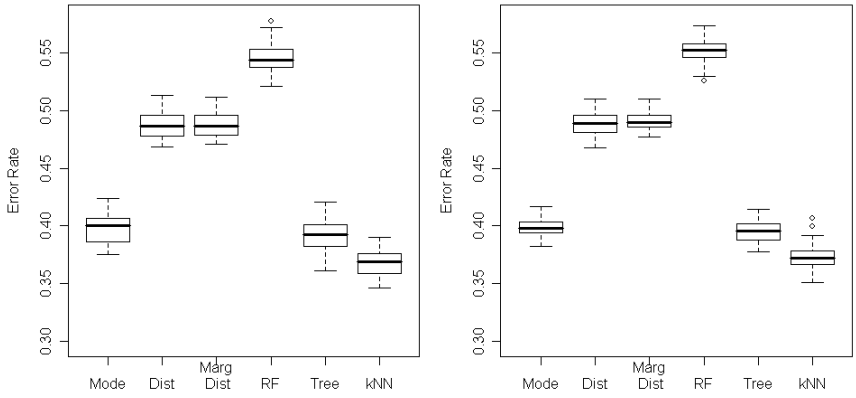

FIGURE B.6. Fractions of falsely imputed genotypes when replacing 5% (left panel) or 10% (right panel) missing values by the mode, by a draw from the SNP-wise distribution, by a draw from the conditional distribution of the SNP given the case-control status, by the Random Forests based method (5 Iterations, 500 trees with 6 SNPs at each node), by the procedure of Dai et al. (2006) with one iteration, and by KNNimpute for categorical data ($k = 50$).

B.2 Supplementary Tables

TABLE B.1. Comparison of the run times of Algorithm 6.4 and the individual calculation of Pearson's χ^2-statistics for differing values of the number m of variables and the number n of observations. Each variable can take $C = 3$ levels, and each observation belongs to one of $R = 2$ classes.

	Algorithm 6.4		Individual	
m	$n = 200$	$n = 1,000$	$n = 200$	$n = 1,000$
50	< 0.01	0.01	0.13	0.16
100	< 0.01	0.02	0.26	0.32
1,000	0.05	0.40	2.64	3.35
10,000	0.63	2.39	26.74	34.42
100,000	6.16	–	274.96	–

TABLE B.2. Parameter values over which the respective discrimination method is optimized in its applications to the data sets used in the comparison of Section 7.4.

Method	Parameter	Values
SVM	Kernel	Linear, Radial, Polynomial
	Costs η	0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 100
	Degree p	2, 3, 4, 5
	$-\log_{10}(\gamma)$	5, 4, 3, 2, 1, 0, -0.30103
PAM	Θ	0.5, 1, 1.5, ..., 12
Bagging	# Iterations (B)	50, 100, 200
Random Forests	# Iterations (B)	500, 1000, 2000, 5000
	# Variables at each Node	$\lfloor\sqrt{m}\rfloor * \begin{bmatrix}0.5 & 1 & 2\end{bmatrix}'$
Logic Regression	Approach	Single, Multiple
	Max. # Leaves	8, 10, 12, 16, 32
	Max. # Trees	2, 3, 4, 5

Appendix C

R Packages

In the following, the R packages containing functions for the procedure introduced in this thesis are presented.

C.1 siggenes

The Significance Analysis of Microarrays (SAM) is implemented in the R package siggenes that can be installed in R by

```
> source("http://www.bioconductor.org/getBioC.R"')
> getBioC("siggenes")
```

Note that this will install the current release version of siggenes, For BioConductor 1.9 released on October 4th, 2006, this is siggenes version 1.8.0 which also contains a function called sam.snp for analyzing categorical data with SAM. The method used in this function is, however, much slower than the approach presented in Section 6.3.3. This new procedure is implemented in siggenes version 1.9.1 and later that can be downloaded from

http://bioconductor.org/packages/2.0/bioc/html/siggenes.html

and will be part of BioConductor 2.0 that will be released in April 2007.

In the following, the help files for the latest version of the function sam.snp, and for the wrapper function sam are shown.

`sam` *Significance Analysis of Microarray*

Description

Performs a Significance Analysis of Microarrays (SAM). It is possible to perform one and two class analyses using either a modified t-statistic or a (standardized) Wilcoxon rank statistic, and a multi-class analysis using a modified F-statistic. Moreover, this function provides a SAM procedure for categorical data such as SNP data.

Usage

```
sam(data, cl, method = "d.stat", delta = NULL, n.delta = 10,
    p0 = NA, lambda = seq(0, 0.95, 0.05), ncs.value = "max",
    ncs.weights = NULL, gene.names = dimnames(data)[[1]],
    q.version = 1, ...)
```

Arguments

`data` a matrix, a data frame, an exprSet, or an ExpressionSet object. Each row of `data` (or `exprs(data)`, respectively) must correspond to a gene, and each column to a sample.

`cl` a vector of length `ncol(data)` containing the class labels of the samples. In the two class paired case, `cl` can also be a matrix with `ncol(data)` rows and 2 columns. If `data` is an exprSet or ExpressionSet object, `cl` can also be a character string naming the column of `pData(data)` that contains the class labels of the samples.

In the one-class case, `cl` should be a vector of 1's.

In the two class unpaired case, `cl` should be a vector containing 0's (specifying the samples of, e.g., the control group) and 1's (specifying, e.g., the case group).

In the two class paired case, `cl` can be either a numeric vector or a numeric matrix. If it is a vector, then `cl` has to consist of the integers between -1 and $-n/2$ (e.g., before treatment group) and between 1 and $n/2$ (e.g., after treatment group), where n is the length of `cl` and k is paired with $-k$, $k = 1, \ldots, n/2$. If `cl` is a matrix, one column should contain -1's and 1's specifying, e.g.,

the before and the after treatment samples, respectively, and the other column should contain integer between 1 and $n/2$ specifying the $n/2$ pairs of observations.

In the multiclass case and if `method="cat.stat"`, `cl` should be a vector containing integers between 1 and g, where g is the number of groups.

For examples of how `cl` can be specified, see the manual of **siggenes**.

`method` a character string specifying the method that should be used in the computation of the expression scores d. If `method = "d.stat"`, a modified t-statistic or F-statistic, respectively, will be computed as proposed by Tusher et al. (2001). If `method = "wilc.stat"`, a Wilcoxon rank sum statistic or Wilcoxon signed rank statistic will be used as expression score. For an analysis of categorical data such as SNP data, `method` can be set to `"cat.stat"`. In this case, Pearson's Chi-squared statistic is computed for each row. It is also possible to use a user-written function to compute the expression scores.

`delta` a numeric vector specifying a set of values for the threshold Δ that should be used. If `NULL`, `n.delta` Δ-values will be computed automatically.

`n.delta` a numeric value specifying the number of Δ values that will be computed over the range of all possible values for Δ if `delta` is not specified.

`p0` a numeric value specifying the prior probability π_0 that a gene is not differentially expressed. If `NA`, `p0` will be computed by the function `pi0.est`.

`lambda` a numeric vector or value specifying the λ values used in the estimation of the prior probability. For details, see `?pi0.est`.

`ncs.value` a character string. Only used if `lambda` is a vector. Either `"max"` or `"paper"`. For details, see `?pi0.est`.

`ncs.weights` a numeric vector of the same length as `lambda` containing the weights used in the estimation of π_0. By default, no weights are used. For details, see `?pi0.est`.

`gene.names` a character vector of length `nrow(data)` containing the names of the genes. By default the row names of `data` are used.

`q.version` a numeric value indicating which version of the q-value should be computed. If `q.version=2`, the original version of the q-value, i.e.

min{pFDR}, will be computed. If `q.version=1`, min{FDR} will be used in the calculation of the q-value. Otherwise, the q-value is not computed. For details, see `?qvalue.cal`.

`...` further arguments of the specific SAM methods. If `method = "d.stat"`, see the help of `sam.dstat`, if `method = "wilc.stat"`, see the help of `sam.wilc`, and if `method = "cat.stat"`, see the help of `sam.snp` for these arguments.

Value

an object of class SAM

Note

SAM was developed by Tusher et al. (2001).

There is a patent pending for the SAM technology at Stanford University.

Author(s)

Holger Schwender

References

Schwender, H., Krause, A., and Ickstadt, K. (2006). Identifying Interesting Genes with `siggenes`. *RNews*, 6(5), 45–50.

Schwender, H., Krause, A., and Ickstadt, K. (2003). Comparison of the Empirical Bayes and the Significance Analysis of Microarrays. *Technical Report*, SFB 475, University of Dortmund, Germany.

Tusher, V.G., Tibshirani, R., and Chu, G. (2001). Significance Analysis of Microarrays Applied to the Ionizing Radiation Response. *PNAS*, 98, 5116-5121.

`sam.snp` *SAM Analysis for Categorical Data*

Description

Performs a SAM (Significance Analysis of Microarrays) analysis for categorical data such a SNP data.

Usage

```
sam.snp(data, cl, B = 100, approx = FALSE, delta = NULL,
  n.delta = 10, p0 = NA, lambda = seq(0, 0.95, 0.05),
  ncs.value = "max", ncs.weights = NULL,
  gene.names = dimnames(data)[[1]], q.version = 1,
  check.levels = TRUE, check.for.NN = FALSE, lev = NULL,
  B.more = 0.1, B.max = 50000, n.subset = 10, rand = NA)
```

Arguments

`data`	a matrix or data frame. Each row must correspond to a variable/SNP, and each column to a sample.
`cl`	a numeric vector of length `ncol(data)` indicating to which class a sample belongs. Must consist of the integers between 1 and C, where C is the number of different groups.
`B`	the number of permutations used in the estimation of the null distribution, and hence, in the computation of the expected d-values. Ignored if `approx=TRUE`.
`approx`	should the null distribution be approximated by the χ^2-distribution?
`delta`	a numeric vector specifying a set of values for the threshold Δ that should be used. If `NULL`, `n.delta` Δ values will be computed automatically
`n.delta`	a numeric value specifying the number of Δ values that will be computed over the range of possible values of Δ if `delta` is not specified.
`p0`	a numeric value specifying the prior probability π_0 that a SNP is not differentially expressed. If `NA`, `p0` will be computed by the function `pi0.est`.
`lambda`	a numeric vector or value specifying the λ values used in the estimation of the prior probability. For details, see the help of `pi0.est`.
`ncs.value`	a character string. Only used if `lambda` is a vector. Either `"max"` or `"paper"`. For details, see the help of `pi0.est`.
`ncs.weights`	a numeric vector of the same length as `lambda` containing the weights used in the estimation of π_0. By default, no weights are used. For details, see the help of `pi0.est`.

`gene.names` a character vector of length `nrow(data)` containing the names of the SNPs. By default, the row names of `data` are used.

`q.version` a numeric value indicating which version of the q-value should be computed. If `q.version=2`, the original version of the q-value, i.e. min{pFDR}, will be computed. If `q.version=1`, min{FDR} will be used in the calculation of the q-value. Otherwise, the q-value is not computed. For details, see `?qvalue.cal`

`check.levels` if `TRUE`, it will be checked if all variables/SNPs have the same number of levels/categories.

`check.for.NN` if `TRUE`, it will be checked if any of the genotypes is equal to "NN". Can be very time-consuming when the data set is high-dimensional.

`lev` numeric or character vector specifying the codings of the levels of the variables/SNPs. Must only be specified if the variables are not coded by the integers between 1 and the number of levels.

`B.more` a numeric value. If the number of all possible permutations is smaller than or equal to (1+`B.more`)*`B`, full permutation will be done. Otherwise, `B` permutations are used.

`B.max` a numeric value. If the number of all possible permutations is smaller than or equal to `B.max`, `B` randomly selected permutations will be used in the computation of the null distribution. Otherwise, `B` random draws of the group labels are used.

`n.subset` a numeric value indicating how many permutations are considered simultaneously when computing the expected d-values.

`rand` numeric value. If specified, i.e. not `NA`, the random number generator will be set into a reproducible state.

Details

For each SNP, Pearson's Chi-Square statistic is computed to test if the distribution of the SNP differs between several groups. Since only one null distribution is estimated for all SNPs as proposed in the original SAM procedure of Tusher et al. (2001), all SNPs must have the same number of levels/categories.

Value

an object of class SAM

Warning

This procedure will only work correctly if all SNPs/variables have the same number of levels/categories.

Note

SAM was developed by Tusher et al. (2001).

There is a patent pending for the SAM technology at Stanford University.

Author(s)

Holger Schwender

References

Schwender, H. (2007). Statistical Analysis of Genotype and Gene Expression Data. *Dissertation*, Department of Statistics, University of Dortmund.

Tusher, V.G., Tibshirani, R., and Chu, G. (2001). Significance Analysis of Microarrays Applied to the Ionizing Radiation Response. *PNAS*, 98, 5116-5121.

C.2 logicFS

The package `logicFS` that can be installed in R by

```
> source("http://www.bioconductor.org/getBioC.R")
> getBioC("logicFS")
```

comprises the following functions for the procedures introduced in Chapter 8.

`logic.fs` *logicFS*

Description

A first basic bagging version of logic regression, and logicFS, a procedure for identifying important interactions of binary variables based on this bagged logic regression. Currently only the classification and the logistic regression approach of `logreg` are available.

Usage

```
logic.bagging(data, cl, B = 100, ntrees = 1, nleaves = 8,
   glm.if.1tree = FALSE, anneal.control = logreg.anneal.control(),
   oob = TRUE, prob.case = 0.5, importance = TRUE, rand = NULL)

logic.fs(data, cl, B = 100, ntrees = 1, nleaves = 8,
   glm.if.1tree = FALSE, anneal.control = logreg.anneal.control(),
   prob.case = 0.5, rand = NULL)
```

Arguments

`data`	a matrix consisting of 0's and 1's. Each column must correspond to a binary variable, and each row to an observation.
`cl`	a vector of 0's and 1's containing the class labels of the observations.
`B`	an integer specifying the number of iterations.
`ntrees`	an integer indicating how many trees should be used. If `ntrees` is larger than 1, the logistic regression approach of logic regreesion will be used. If `ntrees` is 1, then by default the classification approach of logic regression will be used (see `glm.if.1tree`).

`nleaves`	a numeric value specifying the maximum number of leaves used in all trees combined. See the help page of the function `logreg` of the package `LogicReg` for details.
`glm.if.1tree`	if `ntrees` is 1 and `glm.if.1tree` is TRUE the logistic regression approach of logic regression is used instead of the classification approach. Ignored if `ntrees` is not 1.
`anneal.control`	a list containing the parameters for simulated annealing. See `?logreg.anneal.control` of the `LogicReg` package.
`oob`	should the out-of-bag error rate be computed?
`prob.case`	a numeric value between 0 and 1. If the outcome of the logistic regression, i.e. the predicted probability, for an observation is larger than `prob.case` this observations will be classified as case (or 1).
`importance`	should the measure of importance be computed?
`rand`	numeric value. If specified, the random number generator will be set into a reproducible state.

Value

`logic.bagging` returns an object of class `logicBagg` containing

`logreg.model`	a list containing the `B` logic regression models
`inbagg`	a list specifying the `B` Bootstrap samples
`vim`	an object of class `logicFS` (if `importance = TRUE`)
`oob.error`	the out-of-bag error (if `oob = TRUE`)
`...`	further parameters of the logic regression

`logic.fs` returns an object of class `logicFS` containing

`primes`	the prime implicants
`vim`	the importances of the prime implicants
`prop`	the proportions of logic regression models that contain the prime implicants
`type`	the type of model (1: classification, 3: logistic regression)
`param`	further parameters

Author(s)

Holger Schwender

References

Ruczinski, I., Kooperberg, C., and LeBlanc M.L. (2003). Logic Regression. *Journal of Computational and Graphical Statistics*, 12, 475-511.

Schwender, H., Ickstadt, and K. (2006). Identification of SNP Interactions Using Logic Regression. To appear in *Biostatistics*.

`logic.vim` *Variable Importance Measure*

Description

`logic.pimp` computes the prime implicants of an object of class `logicBagg`.

`logic.vim` additionally computes the importances of the prime implicants.

`logic.oob` computes the out-of-bag error of a `logicBagg` object.

Usage

```
logic.pimp(log.out)
logic.oob(log.out, prob.case = 0.5)
logic.vim(log.out, prob.case = 0.5, addInfo = FALSE)
```

Arguments

`log.out`	an object of class `logicBagg`.
`prob.case`	a numeric value between 0 and 1. If the outcome of the logistic regression, i.e. the predicted probability, for an observation is larger than `prob.case` this observations will be classified as case (or 1).
`addInfo`	should further information on the logic regression models be added to the object?

Details

Since we are interested in all potentially interested SNP interactions and not in a minimum set of them, both `logic.pimp` and `logic.vim` return all prime implicants and not a minimum number of them.

Value

`logic.pimp` returns a list consisting of the prime implicants for each of the `B` logic regression models of `log.out`.

`logic.oob` returns the out-of-bag error.

`logic.vim` returns an object of class `logicFS` containing

`primes`	the prime implicants
`vim`	the importances of the prime implicants
`prop`	the proportion of logic regression models that contain the prime implicants
`type`	the type of model (1: classification, 3: logistic regression)
`param`	further parameters (if `addInfo = TRUE`).

Author(s)

Holger Schwender

`make.snp.dummy` *SNPs to Dummy Variables*

Description

Transforms SNPs into binary dummy variables

Usage

```
make.snp.dummy(data)
```

Arguments

`data`	a matrix containing only 1's, 2's, and 3's (see details). Each column of `data` corresponds to a SNP, and each row to an observation.

Details

`make.snp.dummy` assumes that the homozygous reference genotype is coded by 1, the heterozygous genotype by 2, and the homozygous variant genotype by 3. For each SNP, two dummy variables are generated:

SNP_1 At least one of the bases explaining the SNP is the less frequent variant.

SNP_2 Both bases are the less frequent variant.

Value

A matrix with 2*`ncol(data)` columns containing two dummy variables for each SNP.

Author(s)

Holger Schwender

`minDNF` *Minimum Disjunctive Normal Form*

Description

Computes the prime implicants or the minimal disjuntive form, respectively, of a given truth table.

Usage

```
prime.implicants(mat)
minDNF(mat)
```

Arguments

`mat`	a matrix containing only 0's and 1's. Each column of `mat` corresponds to a binary variable and each row to a combination of the variables for which the logic expression is TRUE.

Details

`minDNF` is a fast implementation of the Quine-McCluskey algorithm using matrix algebra.

Value

Either an object of class `minDNF` or of class `primeImp`. Both contain a vector of (a minimum number of) prime implicants. The `primeImp` object additionally contains the prime implicant table.

Author(s)

Holger Schwender

`plot.logicFS` *Variable Importance Plot*

Description

Generates a dotchart of the importances of the most important interactions for an object of class `logicFS` or `logicBagg`.

Usage

```
## S3 method for class 'logicFS':
plot(x, topX = 15, cex = 0.9, pch = 16, col = 1, v0.col = "grey35",
   show.prop = FALSE, force.topX = FALSE, include0 = TRUE,
   coded = TRUE, ...)

## S3 method for class 'logicBagg':
plot(x, topX = 15, cex = 0.9, pch = 16, col = 1, v0.col = "grey35",
   show.prop = FALSE, force.topX = FALSE, include0 = TRUE,
   coded = TRUE, ...)
```

Arguments

`x`	an object of either class `logicFS` or `logicBagg`.
`topX`	integer specifying how many interactions should be shown. If `topX` is larger than the number of interactions contained in `x` all the interactions are shown. For more details, see `force.topX`.
`cex`	a numeric value specifying the relative size of the text and symbols.
`pch`	specifies the used symbol. See `?par` for details.
`col`	the color of the text and the symbols. See `?par` for how colors can be specified.
`v0.col`	the color of the vertical line at $x = 0$. See `?par` for how colors can be specified.
`show.prop`	if `TRUE` the proportions of models that contain the interactions of interest are shown. If `FALSE` (default) the importances of the interactions are shown.

`force.topX` if `TRUE` exactly `topX` interactions are shown. If `FALSE` (default) all interactions up to the `topX`th most important one and all interactions having the same importance as the `topX`th most important one are displayed.

`include0` should $x = 0$ be included in the plot regardless whether the importances of the shown interactions are much higher than 0?

`coded` should the coded variable names be displayed? Might be useful if the actual variable names are pretty long. The coded variable name of the yth variable is "Xy".

`...` Ignored.

Author(s)

Holger Schwender

C.3 c7Tools

The package `c7Tools` contains functions used in this thesis that are not included (at least not in this form) in any other `R` package.

Since some of the functions are modifications of already existing functions, it is planned to publish them in the respective package. Moreover, `c7Tools` will be contributed to the BioConductor project.

As long as it is not published at http://www.bioconductor.org, `c7Tools` can be installed by

```
> url <- "http://www.statistik.uni-dortmund.de/de/content/
+ einrichtungen/lehrstuehle/personen/holgers"
> install.packages("c7Tools",contriburl=url)
```

In the following, the help files of the functions contained in `c7Tools` are presented.

`just.rmaplm` *Preprocessing with RMA Methods*

Description

Computes the signals of RMA and modifications of RMA directly from the CEL files.

Allows to specify if the RMA convolution model or the base composition based methods should be used in the background step, and to select between using median polish or fitting probe level models in the summarization step.

By default, the GCRMA signals are computed.

Usage

```
just.rmaplm(filenames, gc = TRUE, plm = FALSE,
    phenoData = new("phenoData"), description = NULL, notes = "",
    compress = getOption("BioC")$affy$compress.cel,
    affinity.info = NULL, stretch = 1.15 * fast + 1 * (1 - fast),
    type = c("fullmodel", "affinities", "mm", "constant"),
    k = 6 * fast + 0.5 * (1 - fast), correction = 1, rho = 0.7,
```

```
optical.correct = TRUE, verbose = TRUE, fast = TRUE,
minimum = 1, optimize.by = c("speed", "memory"),
model.param = list(), normalize = TRUE, max.its = 20)
```

Arguments

`filenames`	a character vector specifying the names of the CEL files (with path).
`gc`	should the base composition based background method be employed?
`plm`	should robust linear models be fitted in the summarization step?
`phenoData`	a `phenoData` object.
`description`	a `MIAME` object. If `NULL`, a `MIAME` object will be created.
`notes`	character string consisting of notes.
`compress`	are the CEL files compressed?
`affinity.info`	a list consisting of the three components `apm`, `amm` and `index`. If `NULL`, this list will be computed. For details, see the help of `gcrma`.
`stretch`	a tuning parameter.
`type`	a character string naming the type of background correction that should be done if `gc=TRUE`. For details, see the help of `gcrma`.
`k`	a tuning parameter.
`correction`	see help of `gcrma`.
`rho`	correlation coefficient of the log background intensities in a pair of pm/mm probes.
`optical.correct`	should the PMs be corrected for optical noise?
`verbose`	should messages about the progress of the function be printed?
`fast`	should a faster ad hoc algorithm be used? If `TRUE` maximum likelihood estimation, otherwise an empirical Bayes approach is used in the background correction step.
`minimum`	see help of `just.gcrma`.
`optimize.by`	if `"speed"`, a faster algorithm will be used that requires more RAM. If `"memory"`, a slower algorithm will be used that requires less RAM.

`model.param`	a list of parameters controlling the model procedure.
`normalize`	should the probe intensities be quantile normalized?
`max.its`	the number of iterations used in the fit of the probe level model.

Value

An `exprSet` object containing the expression values.

Note

This function might be replaced by the function just3steps in future versions of this package.

Author(s)

Holger Schwender. Based on `just.gcrma` implemented by James W. MacDonald, and on `fitPLM` by Ben Bolstad.

`startPLM` *Fitting PLMs for Large Microarray Experiments*

Description

These functions enable to fit probe level models in experiments comprising hundreds of Affymetrix microarrays.

startPLM performs all three steps of the preprocessing procedure PLM. Instead of startPLM, bgnormPLM can be used to background correct and normalize the probe intensities, and fitLargePLM to summarize these intensities. If startPLM (or one of the other functions) stops because of memory problems, restartPLM can be employed to restart the analysis at the point at which it stopped.

Usage

```
startPLM(filenames, folder = dirname(filenames)[1], mat.xy = NULL,
    batch.size = 1, chunk.size = 100, qn.save = 5, asExprs = TRUE,
    type.save = c("probeset", "both"), digits = 12, max.its = 20,
    save.combine = 100, printDate = TRUE)

bgnormPLM(filenames = NULL, folder = dirname(filenames)[1],
    mat.xy = NULL, batch.size = 1, chunk.size = 100,
```

```
    type.save = c("probeset", "both"), qn.save = 5, digits = 12,
    restart = FALSE)

fitLargePLM(folder, max.its = 20, asExprs = TRUE,
    save.combine = 100, restart = FALSE)

restartPLM(folder)
```

Arguments

`filenames`	a character vector containing the names of the CEL files (with path).
`folder`	a character string naming the folder in which the (temporary) outputs should be stored.
`mat.xy`	a matrix consisting of two columns specifying the x and y coordinates (name of the columns must be `x` and `y`) of the probes that should be contained in the respective probe sets specified by the row names of `mat.xy`. If `mat.xy = NULL`, the cdf environment corresponding to `filenames` will be used. `mat.xy` can be employed to use an alternative cdf environment – not specified by **altcdfenvs** but by a matrix containing the probe set names and the coordinates.
`batch.size`	the number of CEL files read in by `read.probematrix` at once. Recommended: `batch.size = 1`.
`chunk.size`	a numeric value specifying the number of probe sets that are summarized at once. Each chunk of data is saved in one file.
`qn.save`	a numeric value specifying the number of batches after which the vector of the sums of the sorted probe intensities required by quantile normalization is stored in a file. By default, this vector is stored after 5, 10, 15, 20, ... batches of CEL files have been background corrected.
`asExprs`	should the output of `startPLM`, `restartPLM`, or `fitLargePLM` be an `exprSet` object? If `FALSE`, a matrix will be returned.
`type.save`	a character string specifying how the probe intensities are saved. If `type.save = "probeset"`, the probe intensities will be saved in chunks consisting of the intensities from a set of probe sets for all samples. If `type.save = "both"`, the probe intensities will also be stored chip-wisely (in the subfolder bgnorm). Default is

	`"probeset"` which is needed to fit the PLMs specified by `mat.xy` or the cdf environment.
`digits`	a numeric value specifying the number of digits used when storing the chunks of probe intensities. The larger `digits`, the larger are the chunk files. The smaller `digits`, the larger are the differences between the PLM signals obtained by `startPLM` and by `fitPLM`, where these differences are based on the rounding done when saving the chunks.
`max.its`	a numeric value specifying the maximum number of iterations used when fitting the probe level models.
`save.combine`	a numeric value specifying the number of chunks that are combined without storing the combined data set. By default, this data set is stored after combining 100, 200, 300, ... chunks.
`printDate`	should the date when the analysis is started, when, on the one hand, the background correction and the normalization, and on the other hand, the summarization and combining is finished be printed in the `R` window?
`restart`	instead of using the function `restartPLM`, the analysis can also be restarted by setting `restart=TRUE` in either `bgnormPLM` or `fitLargePLM`.

Value

Output of `startPLM`, `restartPLM` and `fitLargePLM` is a matrix or an `exprSet` object depending on the specification of `asExprs`.

Output of `bgnormPLM` is `folder`.

Note

Based on the function `fitPLM` of the package **affyPLM** by Ben Bolstad.

Author(s)

Holger Schwender

`pairs2` *Pairwise Scatter and MA Plots*

Description

Pairwise Scatter and MA Plots with the same ranges for the x and the y axes in each of the plots in either the upper or lower triangle.

`pairs2` generates pairwise scatter plots in the upper triangle, whereas `mva.pairs2` and `smooth.pairs2` generate (smoothed) MA plots in the upper and (smoothed) scatter plots in the lower triangle.

Usage

```
pairs2(x, labels = colnames(x), pch = ".", text.cex = 1.2,
    header = "Scatter Plot", ...)

mva.pairs2(x, labels = colnames(x), span = 2/3,
    family.loess = "gaussian", main = NULL, text.cex = 1.2,
    pch = ".", add.scatter = FALSE, skip.loess = FALSE,
    ab.args = list(col = 3, lwd = 1.5, lty = 1),
    m.args = list(col = 4, lwd = 1.5, lty = 1),
    loess.args = list(col = 2, lwd = 1.5, lty = 1), ...)

smooth.pairs2(x, labels = colnames(x), span = 2/3,
    family.loess = "gaussian", main = NULL, text.cex = 1.2,
    pch = ".", add.scatter = FALSE, skip.loess = FALSE,
    ab.args = list(col = 3, lwd = 1.5, lty = 1),
    m.args = list(col = 4, lwd = 1.5, lty = 1),
    loess.args = list(col = 2, lwd = 1.5, lty = 1), nrpoints = 500,
    colramp = colorRampPalette(c("white",
    RColorBrewer:::brewer.pal(9, "Greys")[-1])))
```

Arguments

`x`	a matrix. The columns of `x` are plotted against each other.
`labels`	a character vector specifying the names for the columns used in the plot.
`pch`	the plotting symbol. For details, see the help for `points`.

`text.cex`	the relative size of the plotted text, i.e. of `labels`.
`header`	a character string specifying the title of the plot.
`span`	the parameter alpha used in the loess fit to control the degree of smoothing.
`family.loess`	either `"gaussian"` or `"symmetric"`. For details, see help of `loess`.
`main`	a character string specifying the title of the plot. If `NULL`, a title will be generated automatically.
`add.scatter`	should scatter plots be added in the lower triangle?
`skip.loess`	should the loess fit be skipped?
`ab.args`	list of graphical arguments for the diagonal in the scatter plots.
`m.args`	list of graphical arguments for the horizontal line at $M = 0$ in the MA plots.
`loess.args`	list of graphical arguments for the loess curves.
`nrpoints`	numeric value specifying the number of points that should be superimposed on the density image.
`colramp`	a function specifying the colors used in the density image.
`...`	further arguments of `plot`.

Note

Modifications of the functions `pairs` and `mva.pairs`.

Author(s)

Holger Schwender

`justPLIER` *Compute PLIER Signals Directly from CEL Files*

Description

These functions generate the unnormalized or quantile normalized PLIER signals directly from the CEL files.

Usage

```
justPLIER(filenames, n.subset = 6,
     compress = getOption("BioC")$affy$compress.cel,
     phenoData = NULL, notes = "", description = NULL,
     replicate = 1:length(filenames), get.affinities = FALSE,
     normalize = FALSE, norm.type = "together",
     samplenames = NULL, ...)

normalize.no.plier(filenames, save.file)

normalize.quantiles.plier(filenames, save.file)

justPLIER2(save.file, n.subset = 6,
     compress = getOption("BioC")$affy$compress.cel,
     phenoData = NULL, notes = "", description = NULL,
     rm.savefile = TRUE, replicate = NULL, get.affinities = FALSE,
     samplenames = NULL, ...)
```

Arguments

`filenames`	a character vector containing the names of the CEL files (with path).
`n.subset`	a numeric value specifying the number of subsets in which the whole set of probe sets should be divided. By default, `n.subset = 6`. This means that if there are, e.g., 60,000 probe sets, 10,000 of them will be processed at once.
`compress`	are the CEL files compressed?
`phenoData`	a `phenoData` object. If `NULL`, a `phenoData` object will be created.
`notes`	character string consisting of notes.
`description`	a `MIAME` object.
`save.file`	a character string ending with `.RData` and naming the file in which the unnormalized or normalized PMs and MMs are stored.
`rm.savefile`	should the file specified by `save.file` be removed after the PLIER signals have been computed?
`replicate`	a factor containing the replicate structure to use for grouping samples.

`get.affinities`	if `TRUE`, then the affinities are returned in the preprocessing slot of the `exprSet` object.
`normalize`	should the PMs and MMs be quantile normalized?
`norm.type`	character string specifying how the PMs and MMs are quantile normalized. Currently, only `together` possible.
`samplenames`	a character string of the same length as `filenames` specifying the names that should be used for the samples in the `exprSet` object. If `NULL`, sample names will be generated from the names of the CEL files.
`...`	further model parameters. See the help of `justPlier`.

Details

`justPLIER` should require much less RAM than the conventional method of generating the PLIER signals by first creating an `AffyBatch` object, and then running `justPlier` on this object.

If there are still memory problems, then `normalize.quantiles.plier` can be used first to generate a file containing the quantile normalized PMs and MMs, and then `justPLIER2` can be employed to compute the PLIER signals based on this file.

Thus, `justPLIER(filenames, normalize = TRUE)` returns the same expression values as a combination of `out <- normalize.quantiles.plier(filenames, save.file)` and `justPLIER2(out)`.

The same applies to the results of `justPLIER(filenames, normalize = FALSE)` and of `justPLIER2(normalize.no.plier(filenames, save.file))`.

Please note that the resulting PLIER signals are already on log2 scale.

Value

For `justPLIER` and `justPLIER2`: An `exprSet` object containing the PLIER signals.

For `normalize.quantiles.plier` and `normalize.no.plier`: `save.file`.

Author(s)

Holger Schwender. Based on the wrapper `justPlier` by Crispin J. Miller, and the C code of Earl Hubbell.

`pamTheta` *Prediction Analysis of Categorical Data*

Description

Adaption of the Prediction Analysis of Microarrays (PAM) to categorical data such as SNPs. Computes the number of variables showing at least one non-zero group-wise shrunken test statistic and the misclassification rate for a set of values for the shrinkage parameter Θ.

Usage

```
pamTheta(data, cl, theta = NULL, n.theta = 10, prep.out = NULL,
   newdata = NULL, check.levels = TRUE, check.for.NN = FALSE)

pamStats(data, cl, check.levels = TRUE, check.for.NN = FALSE)
```

Arguments

`data`	a matrix. Each row must correspond to a variable/SNP, and each column to a sample.
`cl`	a numeric vector of length `ncol(data)` indicating to which class a sample belongs. Must consist of the integers between 1 and C, where C is the number of different groups.
`theta`	a numeric vector consisting of different values for the shrinkage parameter Θ. If `NULL`, `n.theta` values between the minimum and the maximum value of the test statistics are chosen.
`n.theta`	integer specifying the number of considered values of the shrinkage parameter. Ignored if `theta` is specified.
`prep.out`	output of pamStats. If `NULL`, `pamStats` will be called automatically.
`newdata`	the data set used to compute the misclassification rate. If `NULL`, `data` will be used.
`check.levels`	if `TRUE`, it will be checked if all variables/SNPs have the same number of levels/categories.
`check.for.NN`	if `TRUE`, it will be checked if any of the genotypes is equal to "NN". Can be very time-consuming when the data set is high-dimensional.

Author(s)

Holger Schwender

References

Schwender, H. (2007). Statistical Analysis of Genotype and Gene Expression Data. *Dissertation*, Department of Statistics, University of Dortmund.

Tibshirani, R., Hastie, T., Narasimhan, B., and Chu, G. (2002). Diagnosis of Multiple Cancer Types by Shrunken Centroids of Gene Expression. *Proceedings of the National Academy of Sciences*, 99, 6567–6572.

`replace.by.wknn` *Imputing missing categorical data*

Description

Imputes missing categorical data by a procedure based on (weighted) k Nearest Neighbors.

Usage

```
replace.by.wknn(x, dist, nn = 3, samp = FALSE, weights = TRUE)
```

Arguments

`x`	a matrix. Each row must correspond to one of m categorical variables/SNPs, and each column to one of n samples. All variables must have the same number of levels.
`dist`	an $m \times m$ distance matrix. If not specified, the distances are computed based on Pearson's corrected contingency coefficient.
`nn`	integer specifying the number of nearest neighbors considered for the imputation of the missing values.
`samp`	if `FALSE`, then the missing values will be imputed using (weighted) majority voting. If `TRUE`, the (weighted) votes are used as weights for the different levels when drawing the imputed values.
`weights`	should weights based on the distance between the variables be used when imputing the missing values?

Details

Adaption of KNNimpute proposed by Troyanskaya et al. (2001).

Author(s)

Holger Schwender

References

Schwender, H. (2007). Statistical Analysis of Genotype and Gene Expression Data. *Dissertation*, Department of Statistics, University of Dortmund.

Troyanskaya, O.G., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., and Altman, R.B. (2001). Missing Value Estimation Methods for DNA Microarrays. *Bioinformatics*, 17, 520–525.

`rowChisqStat` *Rowwise Pearson's Chi-Square Statistic*

Description

Computes Pearson's Chi-Square Statistic either for testing each pair of rows of a data set if these rows are independent from each other, or for testing each row if the distribution of the variable represented by this row is the same across all groups of observations.

Usage

```
rowChisqStat(data, cl, check = TRUE, asMatrix = TRUE)
```

Arguments

`data`	a matrix or data frame. Each row must correspond to a variable/SNP, and each column to a sample.
`cl`	a numeric vector of length `ncol(data)` indicating to which class a sample belongs. Must consist of the integers between 1 and C, where C is the number of different groups. If specified, each row will be tested if its distribution is the same across all groups of observations specified by `cl`. If not specified, then each pair of rows will be tested if these rows are independent from each other.
`check`	should the data be checked for correctness?

`asMatrix` if the independence test is done, should the test statistics be returned as $m \times m$ matrix, where m is the number of rows? If `FALSE` the lower triangle of this matrix will be returned as vector.

Author(s)

Holger Schwender

`simulateSNPs` *Simulation of SNP data*

Description

Simulates SNP data, where a specified proportion of cases and controls is explained by a specified set of SNP interactions.

Usage

```
simulateSNPs(n.obs, n.snp, vec.ia, prop.explain = 1,
  list.ia.val = NULL, vec.ia.num = NULL, maf = c(0.02, 0.3),
  prob.val = rep(1/3, 3), list.equal = NULL, prob.equal = 0.8,
  rm.redundancy = TRUE, shuffle = FALSE, nosy = TRUE,
  shuffle.obs = FALSE, rand = NA)
```

Arguments

`n.obs` either an integer specifying the total number of observations, or a vector of length 2 specifying the number of cases and the number of controls. If the former, then the number of both cases and controls is `ceiling(n.obs/2)`.

`n.snp` integer specifying the number of SNPs.

`vec.ia` a vector of integers specifying the orders of the interactions that explain the cases. `c(3, 1, 2, 3)`, e.g., means that a three-way, a one-way (i.e. just a SNP), a two-way, and a three-way interaction explain the cases.

`prop.explain` either an integer or a vector of `length(vec.ia)` specifying the proportion of cases explained by the interaction of interest among all observation having the interaction of interest. Must be larger than 0.5. E.g., `prop.explain = 1` means that only cases have the interactions specified by `vec.ia` (and `list.ia.val`). `vec.ia`

`= c(3, 2)` and `prop.explain = c(1, 0.8)` means that only cases have the three-way interaction of interest, while 80% of the observations having the two-way interaction of interest are cases, and 20% are controls.

`list.ia.val` a list of `length(vec.ia)` specifying the exact interactions. The objects in this list must be vectors of `length(vec.ia[i])`, and consist of the values 0 (for homozygous reference), 1 (heterozygous), or 2 (homozygous variant). E.g., `vec.ia = c(3, 2)` and `list.ia.val = list(c(2, 0, 1), c(0, 2))` means that the interactions of interest are SNP1==2 & SNP2==0 & SNP3==1, and SNP4==0 & SNP5==2.

`vec.ia.num` a vector of `length(vec.ia)` specifying the number of *cases* (not observations) explained by the interactions in `vec.ia`. If `NULL`, all the cases are divided into `length(vec.ia)` groups of about the same size. `sum(vec.ia.num)` must be smaller than or equal to the total number of cases. Each entry of `vec.ia.num` must currently be larger than or equal to 10.

`maf` either an integer, or a vector of length 2 or `n.snp` specifying the the minor allele frequency. If an integer, all the SNPs will have the same minor allele frequency. If a vector of length `n.snp`, each SNP will have the minor allele frequency specified in the corresponding entry of `maf`. If length 2, then `maf` is interpreted as the range of the minor allele frequencies, and for each SNP, a minor allele frequency will be randomly drawn from a uniform distribution with the range given by `maf`. Note: If a SNP belongs to an explanatory interaction, then `maf` will only be considered when drawing the genotypes of the observations not explained by this interaction.

`prob.val` a vector consisting of the probabilities for drawing a 0, 1, or 2, if `list.ia.val = NULL`, i.e. if the values of the SNPs explaining the case-control status are randomly drawn. By default, `prob.val = rep(1/3, 3)`.

`list.equal` list of same structure as `list.ia.val` containing only ones and zeros, where a 1 specifies the equality to the corresponding value in `list.ia.val`, and a 0 specifies the non-equality. If `NULL`, this list will be generated automatically using `prob.equal`.

`prob.equal` a numeric value specifying the probability that a 1 is drawn when generating `list.equal`. `prob.equal` is thus the probability for

an equal sign.

`rm.redundancy` should redundant SNPs be removed from the explaining interactions? It is possible that one specify an explaining i-way interaction, but an interaction between $(i-1)$ of the variables in the i-way interaction already explains the cases (and controls) that the i-way interaction should explain. In this case the redundant SNP is removed, if `rm.redundancy = TRUE`.

`shuffle` logical. Usually, the first `sum(vec.ia)` columns of the generated data set contain the explanatory SNPs in the order specified by `vec.ia` and `list.ia.val`. If `TRUE`, this order will be shuffled.

`nosy` logical. If `TRUE`, the explanatory interactions will be displayed (and stored in an object). If `FALSE`, they will only be stored.

`shuffle.obs` should the observations be shuffled?

`rand` integer. Sets the random number generator in a reproducible state.

Author(s)

Holger Schwender

`simulateSNPfblr` *Simulation of SNP interactions*

Description

Simulates SNPs, and randomly draws the case-control status of the observation from a Bernoulli distribution with mean Prob(Y=1), where

$$\text{logit}(\text{Prob}(Y = 1)) = \texttt{beta0} + \texttt{beta} * L_1 + \texttt{beta} * L_2$$

with $L_1 = X_{11}$ AND X_{13}^C, $L_2 = X_5^C$ AND X_{17}^C AND X_{19}^C.

Usage

```
simulateSNPfblr(n.obs = 1000, n.snp = 50, n.cor = 5, beta0 = -1,
   beta = 0.4, sample.y = TRUE, p.cutoff = 0.5)
```

Arguments

`n.obs`	number of observations.
`n.snp`	number of SNPs.
`n.cor`	number of SNPs that should show a correlation structure.
`beta0`	numeric value for the parameter β_0 of the logistic regression.
`beta`	numeric value for the parameter β_i of the logistic regression.
`sample.y`	should the case-control status be sampled? If `FALSE`, each observation with Prob(Y=1) > `p.cutoff` will be called a case.
`p.cutoff`	proportion, i.e. value between 0 and 1. For details, see `sample.y`.

Author(s)

Holger Schwender. Based on code of Arno Fritsch.

Appendix D

Statistical Methods

D.1 MA plot

(*This section is an modified excerpt from Schwender and Belousov, 2006.*)

Alternatively to a scatter plot, a MA plot also known as Bland-Altman plot (Bland and Altman, 1986) can be used to compare, e.g., the signals of two samples or two preprocessing methods. Instead of plotting the ($\log_2$-transformed) expression values directly against each other (as in a scatter plot), the differences of the pairs of ($\log_2$-transformed) values (Minus) are plotted against the

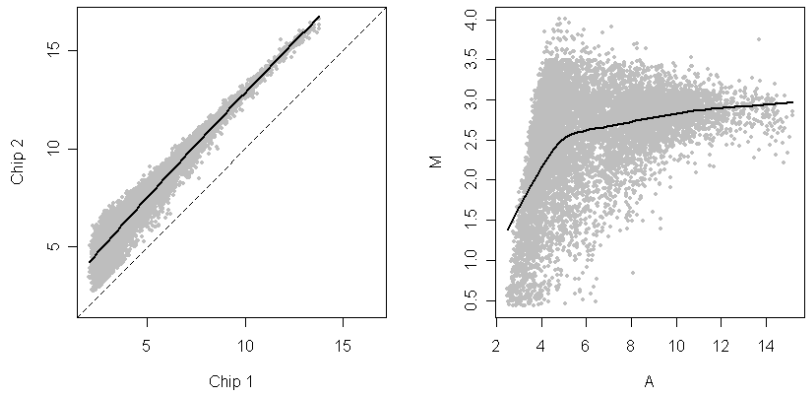

FIGURE D.1. Scatter Plot vs. MA Plot. A subset of the probe intensities of two of the 38 HG-U133_Plus_2 microarrays (cf. Appendix A.1) are plotted against each other using a scatter plot (right panel) and an MA plot (left panel). In both plots, a loess curve (solid line) is fitted through the data points.

averages of these pairs (Add).

Figure D.1 shows the advantages of the MA plot: Non-linearities and overall variability are more clearly visualized than in a scatter plot.

D.2 M-Estimation

Given a vector $\mathbf{y}$ of N observations, an $N \times p$ design matrix $\mathbf{Z}$, and a parameter vector $\boldsymbol{\theta}$ of length p, the linear model $\mathbf{y} = \mathbf{Z}\boldsymbol{\theta} + \boldsymbol{\varepsilon}$ is usually fitted by minimizing the sums of squares of the residuals r_i, $i = 1, \ldots, N$, where $\mathbf{r} = \mathbf{y} - \mathbf{Z}\hat{\boldsymbol{\theta}}$.

Huber (1981) generalizes this approach by estimating $\boldsymbol{\theta}$ by the M-estimator (Maximum likelihood type estimator)

$$\hat{\boldsymbol{\theta}}_{\mathrm{M}} = \min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \rho\left(\frac{r_i}{s}\right) \tag{D.1}$$

for a suitable non-negative function ρ and the scaling parameter s that, e.g., can be estimated by the median absolute deviation (MAD) of the absolute values of the residuals, or by the method of Huber (1981, p. 179ff.).

Since the linear model should be fitted robustly, the deviation from the typical behavior of the observations should be penalized. Therefore, ρ should increase, if the absolute values of the standardized residuals $u_i = r_i/s$ increase.

Minimizing (D.1) leads to solving the p equations

$$\sum_{i=1}^{N} \psi(u_i) z_{ik} = 0, \;\; k = 1, \ldots, p, \tag{D.2}$$

where ψ is the derivative of ρ. If the weights w_i, $i = 1, \ldots, N$, are defined by $w_i = \psi(u_i)/u_i$, then (D.2) becomes

$$\sum_{i=1}^{N} w_i u_i z_{ik} = 0, \;\; k = 1, \ldots, p,$$

which in turn is equivalent to the weighted least squares problem

$$\min \sum_{i=1}^{N} w_i u_i^2.$$

The parameter vector $\boldsymbol{\theta}$ can hence be estimated by

$$\hat{\boldsymbol{\theta}}_{\mathrm{M}} = (\mathbf{Z}'\mathbf{W}\mathbf{Z})^{-1}\,\mathbf{Z}'\mathbf{W}\mathbf{y}, \tag{D.3}$$

where $\mathbf{W}$ is a $N \times N$ diagonal matrix in which the diagonal consists of the weights w_i, $i = 1, \ldots, N$ (see, e.g., Staudte and Sheather, 1990).

To stabilize the estimation of the regression coefficient, (D.3) is not just computed once using, e.g., the residuals of the ordinary least squares regression in the calculation of the weights, but iteratively updated by the IRLS (Iteratively Reweighted Least Squares) procedure described in Algorithm 3.5.

D.3 Statistical Tests

D.3.1 Welch's *t*-Test

The Welch t-statistic

$$t = \frac{\bar{z}_2 - \bar{z}_1}{\sqrt{\dfrac{s_1^2}{n_1} + \dfrac{s_2^2}{n_2}}} \tag{D.4}$$

can be used to test the null hypothesis of equal means in two independent groups of independent observations with unequal variances. Under the assumption of normality, this test statistic is t-distributed with

$$\frac{\left(\dfrac{s_1^2}{n_1} + \dfrac{s_2^2}{n_2}\right)^2}{\dfrac{1}{n_1 - 1}\left(\dfrac{s_1^2}{n_1}\right)^2 + \dfrac{1}{n_2 - 1}\left(\dfrac{s_2^2}{n_2}\right)^2}$$

degrees of freedom.

If any assumption of Welch's t-test is not justifiable, the null distribution of (D.4) can be estimated by a permutation method in which the group labels are repeatedly permuted, and (D.4) is recalculated for each of these permutations.

If the above null hypothesis is tested for m variables, then the p-value of the i^{th} test, $i = 1, \ldots, m$, based on B permutations of the group labels will normally

be determined by

$$p_i = \frac{1}{B} \sum_{b=1}^{B} I\Big(|t_i| \leq \big|t_i^b\big|\Big), \tag{D.5}$$

where t_i is the observed and t_i^b, $b = 1, \ldots, B$, are the permuted statistics of the i^{th} test.

A drawback of (D.5) is that for $B \leq m$, e.g., the Bonferroni adjusted p-values are given by

$$p_i^{\text{adj}} = \min\{mp_i, 1\} = \begin{cases} 0, & \text{if } p_i = 0 \\ 1 & \text{otherwise} \end{cases}.$$

Thus, B should actually be a multiple of m which is impractical if m is huge – as in microarray experiments – or even impossible if m is larger than the total number of permutations.

Storey and Tibshirani (2003a) propose a solution to this problem: Instead of considering each variable/gene individually as in (D.5), they suggest to borrow strength across the genes by assuming that all genes follow the same null distribution, and hence, to compute the i^{th} p-value, $i = 1, \ldots, m$, by

$$p_i = \frac{1}{mB} \sum_{k=1}^{m} \sum_{b=1}^{B} I\Big(|t_i| \leq \big|t_k^b\big|\Big).$$

In Section 4.5, we follow this approach also employed by SAM (see Section 6.3).

D.3.2 Fligner-Killeen Test

The Fligner-Killeen Test can be applied to test the null hypothesis of equal variances in K groups (Conover et al., 1981). The test statistic is computed by setting the j^{th} observation z_{kj}, $j = 1, \ldots, n_k$, in the k^{th} group, $k = 1, \ldots, K$, to $\tilde{z}_{kj} = z_{kj} - \text{median}_k\, z_{kj}$, calculating the rank r_{kj} of $\tilde{z}_{kj}$ in the sample consisting of all $n = \sum_k n_k$ observation, and assigning the normal scores

$$a_n(i) = \Phi^{-1}\left(\frac{i}{2(n+1)} + 0.5\right)$$

to the n observations, where Φ is the standard normal distribution function. The test statistic is then given by

$$\chi^2 = V^{-2} \sum_{k=1}^{K} n_k \left(\bar{A}_k - \bar{a}_n\right)^2,$$

where $\bar{a}_n$ is the mean over $a_n(i)$, $i = 1, \ldots, n$,

$$V^2 = \frac{1}{n-1} \sum_{i=1}^{n} \left(a_n(i) - \bar{a}_n\right), \quad \text{and} \quad \bar{A}_k = \frac{1}{n_k} \sum_{j=1}^{n_k} a_n\left(r_{kj}\right).$$

This test statistic is asymptotically χ^2-distributed with $K-1$ degrees of freedom.

D.3.3 Shapiro-Wilk Test

Shapiro and Wilk (1965) propose a test for normality based on the vector $\boldsymbol{\mu}$ of length n consisting of the expected values of standard normal order statistics for a sample of size n, and the corresponding $n \times n$ covariance matrix $\mathbf{V}$. The Shapiro-Wilk statistic is defined as

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n} \left(x_i - \bar{x}_i\right)^2},$$

where $x_{(1)} \leq \ldots \leq x_{(n)}$ are the n (ordered) observations, and

$$\mathbf{a} = \left(\boldsymbol{\mu}' \mathbf{V}^{-1} \mathbf{V}^{-1} \boldsymbol{\mu}\right)^{-0.5} \boldsymbol{\mu}' \mathbf{V}^{-1}.$$

The p-value is computed by an approximation to the standard normal distribution (for details, see Royston, 1992).

Abbreviations

A	**A**denine
C	**C**ytosine
CART	**C**lassification **A**nd **R**egression **T**rees
DNA	**D**eoxyribo**N**ucleic **A**cid
DNF	**D**isjunctive **N**ormal **F**orm
FDR	**F**alse **D**iscovery **R**ate
FWER	**F**amily-**W**ise **E**rror **R**ate
G	**G**uanine
GENICA	Interdisciplinary study group on **G**ene **EN**vironment **I**nteraction and breast **CA**ncer in Germany
IM	**I**deal **M**ismatch
IQR	**I**nter**Q**uantile **R**ange
kNN	***k*** **N**earest **N**eighbors
logicFS	**logic** **F**eature **S**election
MAD	**M**edian **A**bsolute **D**eviation from the median
MALDI-TOF-MS	**M**atrix **A**ssisted **L**aser **D**esorption/**I**onization – **T**ime **O**f **F**light – **M**ass **S**pectrometry
MAS 5.0	**M**icro**A**rray **S**uite 5.0

MCMC	**M**arcov **C**hain **M**onte **C**arlo
MCR	**M**is**C**lassification **R**ate
MM	**M**is**M**atch
mRNA	**m**essenger **R**ibo**N**ucleic **A**cid
OOB	**O**ut-**O**f-**B**ag
qPLIER	**q**uantile normalized **PLIER**
PAM	**P**rediction **A**nalysis of **M**icroarrays
PCA	**P**rinciple **C**omponent **A**nalysis
PCR	**P**olymerase **C**hain **R**eaction
PLA/PLA+	**P**lier **L**ike **A**lgorithm
PLIER	**P**robe **L**ogarithmic **I**ntensity **ER**ror estimation
PLM	**P**robe **L**evel **M**odel
PM	**P**erfect **M**atch
RAM	**R**andom **A**ccess **M**emory
RFE-SVM	**R**ecursive **F**eature **E**limination using **S**upport **V**ector **M**achines
RMA	**R**obust **M**ulti-array **A**verage
SAM	**S**ignificance **A**nalysis of **M**icroarrays
SNP	**S**ingle **N**ucleotide **P**olymorphism
S/R ratio	**S**ignal-to-**N**oise ratio
SVM	**S**upport **V**ector **M**achine
T	**T**hymine
TPPFP	**T**ail **P**robability for the **P**roportion of **F**alse **P**ositives
VIM	**V**ariable **I**mportance **M**easure

Notations

In the following, the notations are summarized that are used in the same meaning throughout Part II and Part III of this thesis.

General

X_i, S_i, ...	Variables
n, x_{ij}, $PM_{hj}^{(i)}$, ...	Numeric Values
$\mathbf{q}$, ...	Vectors
$\mathbf{X}$, $\mathbf{Z}$, ...	Matrices
$I(\cdot)$, $w(\cdot)$, ...	Functions
$\mathcal{L}$, $\mathcal{S}$, ...	Sets containing, e.g., variables
`R`, `fitPLM`, ...	`R` related objects such as functions, packages, and calls
utility, ...	Directories
http://...	Links to webpages

Variables

L	Logic expression
P_g	Prime implicant
S_i	SNP
S_{i1}, S_{i2}	Dummy variables for SNP S_i
X_i	Explanatory variable
Y	Response

Indices and Numbers

b	Index for permutations
B	Number of permutations
c	Index for columns of a contingency table
C	Number of columns of a contingency table
g	Index of prime implicants
G	Number of prime implicants
h	Index for probes within a probe set
H_i	Number of probes in the i^{th} probe set
i	Index for variables, i.e. probe sets, genes, SNPs
m	Number of variables
j	Index for observations, i.e. samples, microarrays
n	Number of observations
r	Index for rows of a contingency table
R	Number of rows of a contingency table
x, y	The x and the y coordinate of the location of a probe cell on the chip
N_b	Number of correctly classified oob observations in the b^{th} iteration of logicFS
$N_b^{(+g)}$	Number of correctly classified oob observations in the b^{th} iteration of logicFS after adding P_g to the b^{th} logic regression model
$N_b^{(-g)}$	Number of correctly classified oob observations in the b^{th} iteration of logicFS after removing P_g from the b^{th} logic regression model
n_{leaves}	Number of leaves
n_{PM}	Total number of PMs
n_{rc}	Observed number of observations showing level r at the first variable and level c at the second variable of a contingency table

$\tilde{n}_{rc}$	Under the null hypothesis expected number of observations with level r at the first variable and level c at the second variable of a contingency table
$n_{r\cdot}$	Sum over the r^{th} row of a contingency table
$n_{\cdot c}$	Sum over the c^{th} column of a contingency table
n_T	Number of minterms for which the logic expression of interest is true
R	Number of identified genes / rejected null hypotheses
V	Number of false positives

Statistics, Parameter, Probabilities

d_i	Value of the test statistic for variable i
$d^0_{(i)}$	Under the null hypothesis expected value of the i^{th} smallest test statistic
d_{ib}	Value of the test statistic for variable i in permutation b
d_{ir}	Value of the test statistic for variable i in class r
d^{Θ}_{ir}	Value of d_{ir} shrunken by Θ
$IM^{(i)}_{hj}$	Intensity of the ideal mismatch corresponding to $MM^{(i)}_{hj}$
$MM^{(i)}_{hj}$	Intensity of the h^{th} MM in the i^{th} probe set for the j^{th} sample
p_{xy}	Intensity of the probe at location (x, y) on the chip
$PM^{(i)}_{hj}$	Intensity of the h^{th} PM in the i^{th} probe set for the j^{th} sample
S_{hij}	Intensity of the signal of interest measured by the h^{th} probe in the i^{th} probe set for the j^{th} sample
s_0	Fudge factor
s_i	Standard deviation of variable i
t_i	Value of the t-statistic for the i^{th} variable
x_{ij}	Expression value of the i^{th} probe set and the j^{th} sample
$x_{(i)j}$	Column-wisely sorted values such that $x_{(1)j} \leq x_{(2)j} \leq \ldots \leq x_{(n)j}$ for each j

y_j	Value of the response for the j^{th} observation
$\alpha_h^{(i)}$	Effect of the h^{th} probe in the i^{th} probe set
$\beta_j^{(i)}$	Effect of the j^{th} sample for the i^{th} probe set
Δ	Threshold in the Significance Analysis of Microarrays
$\varepsilon_{hj}^{(i)}$	Additive error of the h^{th} PM in the i^{th} probe set for the j^{th} sample when computing the signals for the RMA approaches
$\varepsilon_{hij}^{\text{PM}}, \varepsilon_{hij}^{\text{MM}}$	Multiplicative error of the h^{th} PM/MM in the i^{th} probe set for the j^{th} sample when computing the signals for PLIER and PLA(+)
Θ	Shrinkage parameter in the Prediction Analysis of Microarrays
θ_{kl}	Effect of base $k \in \{\text{A, T, C, G}\}$ at the ℓ^{th} position of the probe sequence
$\lambda^{\text{PM}}, \lambda^{\text{MM}}$	Affinity of the PM/MM probe
$\mu^{(i)}$	Intercept in the i^{th} multi-chip model
π_0	Prior probability that a gene is not differentially expressed
π_r	Prior probability for class r, $r = 1, \ldots, R$
χ_i^2	Pearson's χ^2-statistic for the i^{th} variable
χ_{ir}^2	Pearson's χ^2-statistic for the i^{th} variable in the r^{th} group

Vectors

$\mathbf{1}_n$	Vector of length n containing only ones
$\mathbf{d}$	Vector consisting of the m d_i-values
$\mathbf{d}^0$	Vector comprising the m $d_{(i)}^0$-values
$\mathbf{q}$	Prototype vector for quantile normalization
$\mathbf{y}$	Vector of length n containing the values of the response for the n observations
$\mathbf{x}_{i\cdot}$	Vector of length n consisting of the i^{th} row of $\mathbf{X}$
$\mathbf{x}_j$	Vector of length m comprising the j^{th} column of $\mathbf{X}$

Matrices

$\mathbf{1}_{m,m}$	$m \times m$ matrix composed of ones
$\mathbf{D}^2_{\text{Cont}}$	$m \times m$ matrix containing the squared distance d_{Cont} for each of the $m(m-1)/2$ pairs of variables
$\mathbf{D}^{\text{PAM}}$	$m \times R$ matrix comprising the gene- and group-wise d_{ir}-values
$\mathbf{D}^{\text{Perm}}$	$m \times B$ matrix consisting of the mB permuted d_{ib}-values
$\mathbf{L}$	$B \times n$ matrix containing B permutations of the response $\mathbf{y}$
$\mathbf{L}^{(r)}$	$B \times n$ indicator matrix with elements $l_{bi}^{(r)} = I\left(l_{bi} = r\right)$
$\mathbf{N}^{(rc)}$	$m \times m$ matrix containing the number n_{rc} of observations for each of the $m(m-1)/2$ pairs of variables
$\tilde{\mathbf{N}}^{(rc)}$	$m \times m$ matrix comprising the under the null hypothesis expected number $\tilde{n}_{rc}$ of observations for each of the $m(m-1)/2$ pairs of variables
$\mathbf{T}$	$n_T \times m$ matrix in which each row represents one of the n_T minterms for which the logic expression of interest is true
$\mathbf{X}$	$m \times n$ matrix composed of the values of m variables and n observations
$\mathbf{X}^{(r)}$	$m \times n$ indicator matrix with elements $x_{ij}^{(r)} = I\left(x_{ij} = r\right)$
$\mathbf{Y}$	$m \times R$ matrix in which the r^{th} column indicates if $y_j = r$
$\mathbf{Y}^{(c)}$	$m \times R$ matrix in which the i^{th} row contains the numbers n_{rc} of observations showing class label r and the c^{th} level of the i^{th} variable
$\tilde{\mathbf{Y}}^{(c)}$	$m \times R$ matrix in which the i^{th} row contains the under the null hypothesis expected numbers $\tilde{n}_{rc}$ of observations showing class label r and level c at the i^{th} variable

Functions

$\text{Cont}(\cdot\,,\cdot)$	Pearson's corrected contingency coefficient
$\text{cut}_{\text{low}}(\cdot)$	Lower bound for the rejection region Γ
$\text{cut}_{\text{up}}(\cdot)$	Upper bound for the rejection region Γ

$d_{\cdot}(\cdot\,,\cdot)$	Distance function
$d_{\text{Cont}}(\cdot\,,\cdot)$	Distance function based on Pearson's corrected contingency coefficient
$i(\cdot)$	Impurity function
$I(\cdot)$	Indicator function
$K(\cdot\,,\cdot)$	Kernel function
$SNR(\cdot)$	Signal-to-noise ratio
$T_{\text{B}}(\cdot)$	Tukey's one-step biweight estimate of the mean
$\text{VIM}_{\cdot}(\cdot)$	Variable importance measure
$w(\cdot)$	Weighting function
$\Delta\, i(\cdot\,,\cdot)$	Decrease in impurity
$\Phi(\cdot)$	Standard normal distribution function
$\phi(\cdot)$	Standard normal density function

Miscellaneous

$\mathbf{Z}'$	Transpose of $\mathbf{Z}$
z^{new}	Updated, i.e. background corrected or normalized, value of z
$\lfloor z \rfloor$	Largest integer smaller than or equal to z
Γ	Rejection region
$\mathbb{N}$	Set of natural numbers
Z^C	Complement of Z
$\wedge$	AND-combination/conjunction
$\vee$	OR-combination/disjunction

List of Algorithms

List of Figures

List of Tables

Bibliography

Affymetrix (2001). Affymetrix Microarray Suite. *User's Guide*, Version 5.0, Affymetrix, Santa Clara, CA.

Affymetrix (2002). Statistical Algorithms Description Document. *Technical Report*, Affymetrix, Santa Clara, CA.

Affymetrix (2003). GeneChip® Expression Analysis. *Technical Manual*, Affymetrix, Santa Clara, CA.

Affymetrix (2005). Guide to Probe Logarithmic Intensity Error (PLIER) Estimation. *Technical Note*, Affymetrix, Santa Clara, CA, USA.

Affymetrix (2006). BRLMM: An Improved Genotype Calling Method for the Genechip Human Mapping 500k Array Set. *Technical Report*, Affymetrix, Santa Clara, CA.

Alberts, B., Bray, D., Hopkin, K., Johnson, A., Lewis, J., Raff, M., Roberts, K., and Walter, P. (2005). *Lehrbuch der Molekularen Zellbiologie*. Wiley-VCH, Weinheim, third edition.

Benjamini, Y., and Hochberg, Y. (1995). Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society*, Series B, **57**, 289–300.

Bland, J.M., and Altman, D.G. (1986). Statistical Methods for Assessing Agreement Between two Methods of Clinical Measurement. *Lancet*, **1**, 307–310.

Bolstad, B.M. (2004). Low-Level Analysis of High-Density Oligonucleotide Array Data: Background, Normalization and Summarization. *Dissertation*, University of California, Berkeley, CA.

Bolstad, B.M., Irizarry, R.A., Astrand, M., and Speed, T.P. (2003). A Comparison of Normalization Methods for High Density Oligonucleotide Array Data Based on Variance and Bias. *Bioinformatics*, **19**(2), 185–193.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, **26**, 123–140.

Breiman, L. (2001). Random Forests. *Machine Learning*, **45**, 5–32.

Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. (1984). *Classification and Regression Trees*. Wadsworth, Belmont, CA.

Brown, P.O., and Botstein, D. (1999). Exploring the new World of the Genome with DNA Microarrays. Supplement to *Nature Genetics*, **21**, 33–37.

Büning, H., and Trenkler, G. (1994). *Nichtparametrische Statistische Methoden*. De Gruyter, Berlin, second edition.

Carvalho, B., Speed, T.P., and Irizarry, R.A. (2006). Exploration, Normalization, and Genotype Calls of High Density Oligonucleotide SNP Array Data. *Working Paper* **111**, Department of Biostatistics, Johns Hopkins University, Baltimore, MD.

Cleveland, W.S., and Devlin, S.J. (1988). Locally-Weighted Regression: An Approach to Regression Analysis by Local Fitting. *Journal of the American Statistical Association*, **83**, 596–610.

Conover, W.J., Johnson, M.E., and Johnson, M.M. (1981). A Comparative Study of Tests for Homogeneity of Variances, with Applications to the Outer Continental Shelf Bidding Data. *Technometrics*, **23**(4), 351–361.

Cope, L.M., Irizarry, R.A., Jaffee, H.A., Wu, Z., and Speed, T.P. (2004). A Benchmark for Affymetrix GeneChip Expression Measures. *Bioinformatics*, **20**(3), 323–331.

Cordell, H., and Clayton D. (2002). A Unified Stepwise Regression Procedure for Evaluating the Relative Effects of Polymorphisms Within a Gene Using Case/Control or Family Data: Application to HLA in Type 1 Diabetes. *American Journal of Human Genetics*, **70**, 124–141.

Culverhouse, R., Suarez, B.K., Lin, J., and Reich, T. (2002). A Perspective on Epistasis: Limits of Models Displaying no Main Effect. *American Journal of Human Genetics*, **70**, 461–471.

Dai, J.Y., Ruczinski, I., LeBlanc, M., and Kooperberg, C. (2006) Imputation Methods to Improve Inference in SNP Association Studies. *Genetic Epidemiology*, **30**(8), 690–702.

Dai, M., Wang, P., Boyd, A.D., Kostov, G., Athey, B., Jones, E.G., Bunney, W.B., Myers, R.M., Speed, T.P., Akil, H., Watson, S.J., and Meng, F. (2005). Evolving Gene/Transcript Definitions Significantly Alter the Interpretation of GeneChip Data. *Nucleic Acids Research*, **33**(20), e175.

Diaz-Uriarte, R., and Alvarez de Andres, S. (2006). Gene Selection and Classification of Microarray Data Using Random Forest. *BioMed Central Bioinformatics*, **7**:3.

Dudoit, S., Shaffer, J. P., and Boldrick, J.C. (2003). Multiple Hypothesis Testing in Microarray Experiments. *Statistical Science*, **18**(1), 71–103.

Efron, B., and Tibshirani, R. (2002). Empirical Bayes Methods and False Discovery Rates for Microarrays. *Genetic Epidemiology*, **23**, 70–86.

Efron, B., Tibshirani, R., Storey, J.D., and Tusher, V. (2001). Empirical Bayes Analysis of a Microarray Experiment. *Journal of the American Statistical Association*, **96**, 1151–1160.

Eschrich, S., Yang, I., Bloom, G., Kwong K.Y., Boulware, D., Cantor, A., Coppola, D., Kruhoffer, M., Aaltonen, L., Orntoft T.F., Quackenbush, J., and Yeatman, T.J. (2005). Molecular Staging for Survival Prediction of Colorectal Cancer Patients. *Journal of Clinical Oncology*, **23**(15), 3526–3535.

Fix, E., and Hodges, J.L. (1951), Discriminatory Analysis – Nonparametric Discrimination: Consistency Properties. *Technical Report*, USAF School of Aviation Medicine.

Freund, Y., and Shapire, R.E. (1996). Experiments with a new Boosting Algorithm. In *Machine Learning: Proceedings of the 13th International Conference*, Morgan Kauffman, San Francisco, CA, 148–156.

Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive Logistic Regression: A Statistical View of Boosting. *Annals of Statistics*, **28**, 337–407, with discussion.

Fritsch, A. (2006). A Full Bayesian Version of Logic Regression for SNP Data. *Diploma Thesis*, Department of Statistic, University of Dortmund.

Garte, S. (2001). Metabolic Susceptibility Genes as Cancer Risk Factors: Time for a Reassessment? *Cancer Epidemiology, Biomarkers and Prevention*, **10**, 1233–1237.

Ge, Y., Dudoit, S., and Speed, T.P. (2003). Resampling-Based Multiple Testing for Microarray Data Analysis. *TEST*, **12**, 1–44, with discussion: 44–77.

Gelman, A., Carlin, J.B., Stern, H.S., and Rubin, D.B. (2003). *Bayesian Data Analysis*, Chapman & Hall, London, second edition.

Genovese, C., and Wasserman, L. (2002). Operating Characteristics and Extensions of the Procedure. *Journal of the Royal Statistical Society*, Series B, **64**, 499–517.

Gentleman, R.C., Carey, V.J., Bates, D.M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., Huber, W., Iacus, S., Irizarry, R., Leisch, F., Li, C., Maechler, M., Rossini, A.J., Sawitzki, G., Smith, C., Smyth, G., Tierney, L., Yang, J.Y.H., and Zhang, J. (2004). Bioconductor: Open Software Development for Computational Biology and Bioinformatics. *Genome Biology*, **5**, R80.

Gentleman, R.C., Carey, V.J., Huber, W., Irizarry, R.A., and Dudoit, S., eds. (2005). *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, Springer, New York, NY.

Gonick, L., and Wheelis, M. (1991). *The Cartoon Guide to Genetics*. Harper Perennial, New York, NY, updated edition.

Greenland, S., and Finkle, W. (1995). A Critical Look at Methods for Handling Missing Covariates in Epidemiologic Regression Analysis. *American Journal of Epidemiology*, **142**, 1255–1264.

Guyon, I., Gunn, S., Nikravesh, M., and Zadeh, L.A., eds. (2006). *Feature Extraction. Foundations and Applications*. Springer, Heidelberg.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene Selection for Cancer Classification Using Support Vector Machines. *Machine Learning*, **46**, 389–422.

Hastie, T., Tibshirani, R., and Friedman, J.H. (2001). *The Elements of Statistical Learning*. Springer, New York, NY.

Heidema, G.A., Boer, J.M.A., Nagelkerke, N., Mariman, E.C.M., A, D.L. van de, and Feskens, E.J.M. (2006). The Challenge for Genetic Epidemiologists: How to Analyze Large Numbers of SNPs in Relation to Complex Diseases. *BioMed Central Genetics*, **7**:23.

Hekstra, D., Taussig, A.R., Magnasco, M., and Naef, F. (2003). Absolute mRNA Concentrations from Sequence-Specific Calibration of Oligonucleotide Arrays. *Nucleic Acids Research*, **31**(7), 1962–1968.

Huber, P.J. (1981). *Robust Statistics*. Wiley, New York, NY.

Ickstadt, K., Bernholt, T., Fritsch, A., Nunkesser, R., Schwender, H., and Fried, R. (2006a). MCMC or Genetic Programming? Comparing Search Algorithms for the Analysis of Genotype Data. *Poster*, ISBA 8$^{\text{th}}$ World Meeting on Bayesian Statistics, Benidorm, Spain.

Ickstadt, K., Müller, T., and Schwender, H. (2006b). Analyzing SNPs: Are There Needles in the Haystack? *Chance*, **19**(3), 21–26.

Ihaka, R., and Gentleman, R. (1996). R: A Language for Data Analysis and Graphics. *Journal of Computational and Graphical Statistics*, **5**, 299–314.

Irizarry, R.A., Bolstad, B.M., Collin, F., Cope, L.M., Hobbs, B., and Speed, T.P. (2003). Summaries of Affymetrix GeneChip Probe Level Data. *Nucleic Acids Research*, **31**(4), e15.

Irizarry, R.A., Wu, Z., and Jaffee, H.A. (2006). Comparison of Affymetrix GeneChip Expression Measures. *Bioinformatics*, **22**(7), 789–794.

Johnson, R.A., and Wichern, D.W. (1998). *Applied Multivariate Statistical Analysis*. Prentice Hall, Upper Saddle River, NJ, fourth edition.

Jung, K. (2006). Contributions to Statistical Techniques for the Analysis of Gene and Protein Expression Data. *Dissertation*, Department of Statistics, University of Dortmund.

Justenhoven, C., Hamann, U., Pesch, B., Harth, V., Rabstein, S., Baisch, C., Vollmert, C., Illig, T., Ko, Y., Brüning, T., and Brauch, H. (2004). ERCC2 Genotypes and a Corresponding Haplotype are Linked with Breast Cancer Risk in a German Population. *Cancer Epidemiology, Biomarkers and Prevention*, **13**, 2059–2064.

Justenhoven, C., Hamann, U., Schubert, F., Zapatka, M., Pierl, C., Rabstein, S., Selinski, S., Müller, T., Ickstadt, K., Gilbert, M., Ko, Y., Baisch, C., Pesch, B., Harth, V., Bolt, H.M., Vollmert, C., Illig, T., Eils, R., Dippon, J., and Brauch, H. (2006). Breast Cancer: A Candidate Gene Approach Across the Estrogen Metabolic Pathway. Submitted to the *Journal of the National Cancer Institute*.

Kennedy, G.C., Matsuzaki, H., Dong, S., Liu W., Huang, J., Liu, G., Su, X., Cao, M., Chen, W., Zhang, J., Liu, W., Yang, G., Di, X., Ryder, T., He, Z., Surti, U., Phillips, M.S., Boyce-Jacino M.T., Fodor S.P., and Jones, K.W. (2003). Large-Scale Genotyping of Complex DNA. *Nature Biotechnology*, **21**, 1233–1237.

Kirkpatrick, S., Gelatt, C.D. Jr., and Vecchi, M.P. (1983). Optimization by Simulated Annealing. *Science*, **220**, 671–680.

Klose, J., and Kobalz, U. (1995). Two-Dimensional Electrophoresis of Proteins: An Updated Protocol and Implications for a Functional Analysis of the Genome. *Electrophoresis*, **16**, 1034–1059.

Kooperberg, C., and Ruczinski, I. (2005). Identifying Interacting SNPs Using Monte Carlo Logic Regression. *Genetic Epidemiology*, **28**, 157–170.

Kooperberg, C., Ruczinski, I., LeBlanc, M., and Hsu, L. (2001). Sequence Analysis Using Logic Regression. *Genetic Epidemiology*, **21**, 626–631.

Kuhn, K., Baker, S.C., Chudin, E., Lieu, M.H., Oeser, S., Bennett, H., Rigault, P., Barker, D., McDaniel, T.K., and Chee, M.S. (2004). A Novel, High-Performance Random Array Platform for Quantitative Gene Expression Profiling. *Genome Research*, **14**, 2347–2356.

Laan, M.J. van der, Birkner, M.D., and Hubbard, A.E. (2005). Resampling Based Multiple Testing Procedure Controlling Tail Probability of the Proportion of False Positives. *Statistical Applications in Genetics and Molecular Biology*, **4**, Article 29.

Laan, M.J. van der, Dudoit, S., and Pollard, K.S. (2004). Augmentation Procedures for Control of the Generalized Family-Wise Error Rate and Tail Probabilities for the Proportion of False Positives. *Statistical Applications in Genetics and Molecular Biology*, **3**, Article 15.

Lipshutz, R.J., Fodor, S.P.A., Gingeras, T.R., and Lockhart, D.J. (1999). High Density Synthetic Oligonucleotide Arrays. Supplement to *Nature Genetics*, **21**, 20–24.

Little, R.J.A., and Rubin, D.B. (1987). *Statistical Analysis with Missing Data*. Wiley, New York, NY.

Liu, W. (2004). Documentation to GX. *Technical Note*, Roche Molecular Systems, Alameda, CA.

Lönnstedt, I., and Speed, T.P. (2002). Replicated Microarray Data. *Statistica Sinica*, **12**, 31–46.

McCluskey, E. (1956). Minimization of Boolean Functions. *Bell System Technical Journal*, **35**, 1417–1444.

Müller, T., Selinski, S., and Ickstadt, K. (2005). Cluster Analysis: A Comparison of Different Similarity Measures for SNP Data. *Technical Report*, SFB 475, Department of Statistics, University of Dortmund.

Naef, F., and Magnasco, M.O. (2003). Solving the Riddle of the Bright Mismatches: Labeling and Effective Binding in Oligonucleotide Arrays. *Physical Review E*, **68**, 011906.

Neter, J., Kutner, M.H., Nachtsheim, C.J., and Wasserman, W. (1996). *Applied Linear Statistical Models*, McGraw-Hill/Irwin, Chicago, IL, fourth edition.

Pudil, P., Novovicova, J., and Kittler, J. (1994). Floating Search Methods in Feature Selection. *Pattern Recognition Letters*, **15**(11), 1119–1125.

Pusch, W., Wurmbach, J.H., Thiele, H., and Kostrzewa, M. (2002). MALDI-TOF Mass Spectrometry-Based SNP Genotyping. *Pharmacogenomics*, **3**(4), 537–548.

Quine, W.V. (1952). The Problem of Simplify Truth Functions. *American Mathematical Monthly*, **59**(8), 521–531.

Rabbee, N., and Speed, T.P. (2006). A Genotype Calling Algorithm for Affymetrix SNP Arrays. *Bioinformatics*, **22**(1), 7–12.

Rabe, C. (2004). Identifying Interactions in High Dimensional SNP Data Using MDR and Logic Regression. *Diploma Thesis*, Department of Statistics, University of Dortmund.

Ripley, B.D. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge.

Ritchie, M.D., Hahn, L.W., Roodi, N., Bailey, L.R., Dupont, W.D., Parl, F.F., and Moore, J.H. (2001). Multifactor-Dimensionality Reduction Reveals High-Order Interactions Among Estrogen-Metabolism Genes in Sporadic Breast Cancer. *American Journal of Human Genetics*, **69**, 138–147.

Royston, P. (1992). Approximating the Shapiro-Wilk W-Test for Non-Normality. *Statistics and Computing*, **2**, 117–119.

Ruczinski, I. (2000). Logic Regression and Statistical Issues Related to the Protein Folding Problem. *Dissertation*, Department of Statistics, University of Washington, Seattle, WA.

Ruczinski, I., Kooperberg, C., and LeBlanc, M. (2003). Logic Regression. *Journal of Computational and Graphical Statistics*, **12**, 475–511.

Ruczinski, I., Kooperberg, C., and LeBlanc, M. (2004). Exploring Interactions in High-Dimensional Genomic Data: An Overview of Logic Regression, with Applications. *Journal of Multivariate Analysis*, **90**, 178–195.

Schmitt, R.I. (2005). Logic Regression in Diagnostic Classification Problems. *Diploma Thesis*, Department of Statistics, University of Dortmund.

Schwender, H. (2003). Assessing the False Discovery Rate in a Statistical Analysis of Gene Expression Data. *Diploma Thesis*, Department of Statistics, University of Dortmund.

Schwender, H. (2005). Modifying Microarray Analysis Methods for Categorical Data – SAM and PAM for SNPs. In Weihs, C., and Gaul, W. (eds.), *Classification – The Ubiquitous Challenge*. Springer, Heidelberg, 370–377.

Schwender, H. (2007). Minimization of Boolean Expressions Using Matrix Algebra. *Technical Report*, Department of Statistics, University of Dortmund.

Schwender, H., and Belousov, A. (2006). Comparison of Preprocessing Methods for Affymetrix Microarrays. *Chance*, **19**(3), 15–20.

Schwender, H., and Ickstadt, K. (2007). Identification of SNP Interactions Using Logic Regression. To appear in *Biostatistics*.

Schwender, H., Krause, A., and Ickstadt, K. (2006a). Identifying Interesting Genes with `siggenes`. *RNews*, **6**(5), 45–50.

Schwender, H., Rabstein, S., and Ickstadt, K. (2006b). Do you Speak Genomish? *Chance*, **19**(3), 3–8.

Schwender, H., Zucknick, M., Ickstadt, K., and Bolt, H.M. (2004). A Pilot Study on the Application of Statistical Classification Procedures to Molecular Epidemiological Data. *Toxicology Letters*, **151**, 291–299.

Selinski, S., and Ickstadt, K. (2005). Similarity Measures for Clustering SNP Data. *Technical Report*, SFB 475, Department of Statistics, University of Dortmund.

Shaffer, J.P. (1995). Multiple Hypothesis Testing. *Annual Review of Psychology*, **46**, 561–584.

Shapiro, S.S., and Wilk, M.B. (1965). An Analysis of Variance Test for Normality (Complete Samples). *Biometrika*, **52**, 591–611.

Smyth, G.K. (2004). Linear Models and Empirical Bayes Methods for Assessing Differential Expression in Microarray Experiments. *Statistical Applications in Genetics and Molecular Biology*, **3**(1), Article 3.

Somol, P., Pudil, P., Novovicova, J., and Paclik, P. (1999). Adaptive Floating Search Methods in Feature Selection. *Pattern Recognition Letters*, **20**, 1157–1163.

Stange, D.E., Radlwimmer, B., Schubert, F., Traub, F., Pich, A., Toedt, G., Mendrzyk, F., Lehmann, U., Eils, R., Kreipe, H., and Lichter, P. (2006). High-Resolution Genomic Profiling Reveals Association Chromosomal Aberrations on 1q and 16p with Histologic and Genetic Subgroups of Invasive Breast Cancer. *Clinical Cancer Research*, **12**, 345–352.

Staudte, R.G., and Sheather, S.J. (1990). *Robust Estimation and Testing*. Wiley, New York, NY.

Storey, J.D. (2002). A Direct Approach to False Discovery Rates. *Journal of the Royal Statistical Society*, Series B, **64**, 479–498.

Storey, J.D., and Tibshirani, R. (2003a). SAM Thresholding and False Discovery Rates for Detecting Differential Gene Expression in DNA Microarrays. In Parmi-

giani, G., Garrett, E.S., Irizarry, R.A., and Zeger, S.L. (eds.), *The Analysis of Gene Expression Data: Methods and Software*. Springer, New York, NY, 272–290.

Storey, J.D., and Tibshirani, R. (2003b). Statistical Significance for Genome-Wide Studies. *Proceedings of the National Academy of Sciences*, **100**, 9440–9445.

Strachan, T., and Read, A.P. (2005). *Molekulare Humangenetik*. Spektrum Akademischer Verlag, Munich, third edition.

Sydor, J.R., and Nock, S. (2003). Protein Expression Profiling Arrays: Tools for the Multiplexed High-Throughput Analysis of Proteins. *BioMed Central Protein Science*, **1**:3.

The International HapMap Consortium (2003). The International HapMap Project. *Nature*, **426**, 789–796.

Tibshirani, R., Hastie, T., Narasimhan, B., and Chu, G. (2002). Diagnosis of Multiple Cancer Types by Shrunken Centroids of Gene Expression. *Proceedings of the National Academy of Sciences*, **99**, 6567–6572.

Troyanskaya, O.G., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., and Altman, R.B. (2001). Missing Value Estimation Methods for DNA Microarrays. *Bioinformatics*, **17**, 520–525.

Tukey, J.W. (1977). *Exploratory Data Analysis*. Addison-Wesley, Reading, MA.

Tusher, V., Tibshirani, R., and Chu, G. (2001). Significance Analysis of Microarrays Applied to the Ionizing Radiation Response. *Proceedings of the National Academy of Sciences*, **98**, 5116–5124.

Vapnik, V. (2000). *The Nature of Statistical Learning Theory*. Springer, New York, NY, second edition.

Veer, L.J. van 't, Dai, H, Vijver, M.J. van de, He, Y.D., Hart, A.A., Mao, M., Peterse, H.L., Kooy, K. van der, Marton, M.J., Witteveen, A.T., Schreiber, G.J., Kerkhoven, R.M., Roberts C., Linsley, P.S., Bernards, R., and Friend, S.H. (2002). Gene Expression Profiling Predicts Clinical Outcome of Breast Cancer. *Nature*, **415**, 530–536.

Wand, M.P., and Jones, M.C. (1995). *Kernel Smoothing*. Chapman and Hall, London.

Wang Y., Jatkoe, T., Zhang, Y., Mutch, M.G., Talantov, D., Jiang, J., McLeod H.L., and Atkins, D. (2004). Gene Expression Profiles and Molecular Markers to Predict Recurrence of Dukes' B Colon Cancer. *Journal of Clinical Oncology*, **22**(9), 1564–1571.

Westfall, P.H., and Young, S.S. (1993). *Resampling-Based Multiple Testing: Examples and Methods for P-Value Adjustments*. Wiley, New York, NY.

Wichert, S., Fokianos, K., and Strimmer, K. (2004). Identifying Periodically Expressed Transcripts in Microarray Time Series Data. *Bioinformatics*, **20**, 5–20.

Witte, J.S., and Fijal, B.A. (2001). Introduction: Analysis of Sequence Data and Population Structure. *Genetic Epidemiology*, **21**, 600–601.

Wu, Z., Irizarry, R.A., Gentleman, R., Martinez-Murillo, F., and Spencer, F. (2004). A Model-Based Background Adjustment for Oligonucleotide Expression Arrays. *Journal of the American Statistical Association*, **99**, 909–917.