Low-level analysis of microarray data

Bengtsson, Henrik

2004

Low-level Analysis of Microarray Data

Henrik Bengtsson

Lund Institute of Technology
Lund University

Centre for Mathematical Sciences
Mathematical Statistics

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118
SE-221 00 Lund
Sweden

http://www.maths.lth.se/

Doctoral Theses in Mathematical Sciences 2004:6
ISSN 1404-0034

ISBN 91-628-6215-4
LUTFMS-1024-2004

Printed in Sweden by KFS AB, Lund 2004

Contents

Paper F:

Paper G:

List of papers

The thesis consists of the following papers listed in chronological order:

A) H. Bengtsson, B. Calder, I. S. Mian, M. Callow, E. Rubin, and T. P. Speed. Identifying Differentially Expressed Genes in cDNA Microarray Experiments: making aging visible. Online report accompanying the discussion forum with the same name. Science SAGE KE, 2001 (12), vp8.

B) H. Bengtsson. Identification and normalization of plate effect in cDNA microarray data. Preprints in Mathematical Sciences 2002:28, Mathematical Statistics, Centre for Mathematical Sciences, Lund University, 2002.

C) H. Bengtsson. The R.oo package - object-oriented programming with references using standard R code. In Kurt Hornik, Friedrich Leisch, and Achim Zeileis, editors, *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*, Vienna, Austria, March 2003.

D) H. Bengtsson and O. Hössjer. Methodological study of affine transformations of gene expression data with proposed normalization method. Preprints in Mathematical Sciences 2003:38, Mathematical Statistics, Centre for Mathematical Sciences, Lund University, 2003. [submitted]

E) H. Bengtsson, G. Jönsson, and J. Vallon-Christersson. Calibration and assessment of channel-specific biases in microarray data with extended dynamical range. Preprints in Mathematical Sciences 2003:37, Mathematical Statistics, Centre for Mathematical Sciences, Lund University, 2003. [tentatively accepted for BMC Bioinformatics]

F) H. Bengtsson. aroma - An R Object-oriented Microarray Analysis environment. Preprints in Mathematical Sciences 2004:18, Mathematical Statistics, Centre for Mathematical Sciences, Lund University, 2004.

G) H. Bengtsson and O. Hössjer. Affine calibration for microarrays with dilution series or spike-ins. Preprints in Mathematical Sciences 2004:19, Mathematical Statistics, Centre for Mathematical Sciences, Lund University, 2004. [submitted]

Acknowledgments

There are so many people I would like to thank for making it possible for me to enter the wonderful world of science. The list of names and reasons can be made very long - now I will not say that I do not have the space to thank everyone, because I do - and therefore I will try to do it. If someone is not mentioned it is not because I have forgotten that person, but because I wrote these acknowledgments just before sending this thesis to print.

I wish to thank my co-supervisor Jan Holst for taking me on as a Master's Thesis student and later as a PhD student. You inspired and supported me whenever needed. I also wish to thank my supervisor Ola Hössjer for showing me how problems should really be solved. I have learned so much from you, but still so little. Thanks for everything. I am also in great debt to Terry Speed who accepted me as a student while I was at UC Berkeley and later for inviting me to WEHI. Thanks for giving me (and many others) the opportunity to experience the great world of microarrays and science in general. I will always carry the advice that "you have to know your data" with me.

Thanks to Peter Karcher for introducing me to Bayes Nets, which became the topic of my Master's project and my first year of my PhD studies. Thanks to Jacob for all the great flying hours over California, Nevada and Denmark. Thanks to all my current and former colleagues in Lund - your all are part of this and made my past years a great trip - Sofia A, Fredrik, Björn, Roger, Halfdan (thanks a lot for the PCA discussions), Finn (to you too), Anna, Anastassia, Sofia Å, Ulla, Torgny, Aurelia, Jesper, Magnus, Dragi, Linda, Martin, Sebastian, Lars, Sara, Thomas, Erik, Mona (!), Mats, Georg, Svetlana, Klas, Peter, James (!), Attila, Jörgen, Jimmy, Anders, Johan, Åsa, Joakim (!), Bengt, Jakob, Igor, Jens, Lena, Carl-Gustav, Jan, Tatyana, Tobias, Oskar, Maria, Ulla, Anders, Pia, Nader, Mikael, Søren, Pär-Ola, Lars, Roland, Anders, Azra! Thanks also to Peter Guttorp for last minute grammar help. Ola, `119^4>1 61^14>6 206^4>8 99_9>2 84^17<2 12_7>6 6^11>4; 188_14<4 189_18<2 225^5>3 225^5<5 3^6>7 3^6>8. 77^7>1 163_2>1 232_7<2 218_13<5!` My great friend of life Björn and wife Sandra, see you soon and more often. Pete, Daniel, and Roger, we have to pick up where we ended, many more things to be discussed and solved. I still remember my great math classes in school - thanks Kjell Sörhede and Bo Henriksson. Thanks also Per-Anders Ivert for that great Mathematics A course you gave in 1991. Thanks to Fritjof and Leena for hiking, friendship and support. What would life have been without neighbors Geoff and Jody on 6690 Abrego Road? Thanks to Lars and Catarina - long time no see - maybe next week when I've filed my thesis? Azra, you're a great friend! Patrik and Nuno, we should really pick up our ski trips again. Toby in Toronto, you're great! Thanks Guilem for trying to teach me Portuguese and passing on the secrets of life! If it would not have been for the great coffee and warm people at Brewed Awakening, Thresherman's, Ariman and Mondo, this thesis would have been - you can't imaging what great ideas you get in cafes like these (true). Ann-Louise you're eternally the greatest. Johan for being a great friend and room mate on 6690 and now also a great research mate; we have a lot more to discover. If there is a word like "spot spotter", I dedicate it to Göran. Dr Depken, love you dude! Thanks Marina (did you ever fix that final bug?), Hannes (for everything!), Michael, Angelika, and Dan for being the best neighbors you can ever get. There is only one place like 150 Panoramic Way and that's because of you. Gara, thanks for being patient. Thanks Cal Ski Team. Jonny and Anna-Maria, I challenge you on skiing this winter. Are you in? Jon and Andrea, you made my year and thanks for sharing all of your wonderful friends. Dave, when your back from Galapagos I hope we can catch up on the spicy chicken - but please no more singing in the convent. Mia, it was fun. Dear co-movie star Xiaoyue, would you care for another 11-0 game? Thanks Sandrine, Jean, and Ben for your guidance when I started out my microarray work. Thomas, we will get that beer we always talk about, ehe? It was a great time Fernanda and Raquel. I'm glad that we met Jessica. Ingileif and Lior, see you soon? Javier and Mariel you inspire me in science and life - we should do more clubbing at Liquid too. Thanks director Sally and Terry for your hospitality. Thanks Tom for being a great flatmate. Tim, Gordon, and Asa, thanks for all great dinners and everything else. James, Melanie, Asa, Matt, Keith, Frédéric, Natalie, Alex, Suzanne, Ken, Rusell, Chris, Lavinia, Penny, Kate, Eleanor, Adele, and all others at WEHI - thanks for making my visit the best - hope to see you all soon. Andreia and Marlos, I promised myself to focus 100% on research during my visit to Melbourne, but that last about 30 minutes until I ran into you guys. Our trip to Tasmania with Rodgrigo, Veronica, Takashi and Mark was "heaps of fun". Ewa, I will always remember that spontaneous night out the day before Xmas eve! Jane the prayer flag killer, you're wonderful. Thanks David for being a great friend. The Centre for Mathematical Sciences' football team, here we go! Thanks to Johan and Björn for comments on the introduction of this thesis. My courtesy to the R community. Bless my lucky star for providing me with an extremely perfect timing in order for everything to happen at the right time and right place. Finally, I would like to thank my parents Anna-Lena and Gert and Fredrik and Ulrika for always being there (the next 250 pages is what I've up to all this time). Thanks you all for being beautiful people.

Lund (Mondo), August 26, 2004.

Henrik Bengtsson

Low-level analysis of microarray data

-That's one small step for mankind; one giant leap for a man.

1 Introduction

In today's life sciences much attention is on systems biology and functional genomics where researchers try to identify and study important genes and complex genetic regulatory networks in order to understand the fundamental properties of living organisms and especially the dynamics of the cells. This thesis is all about cleaning up measurements obtained by one very promising and less than a decade old technology, a technology that makes it possible for scientists to study the cells in ways that have never been done before.

The introductory part of this seven-paper PhD thesis in Mathematical Statistics is outlined as follows. In Section 2, we give a brief introduction to the molecular biology of the cell. In the following section we introduce *the* marvelous technology, explain how it works, and give some initial motivations for the area of our own work. In Section 4, in order to further motivate and give more background to the papers presented, we give our formalized view of gene-expression analysis in general. In Section 5, we describe the software developed making it possible for us to quickly communicate new statistical methods with the research community. In the final section, based on these introductory sections, we conclude by providing a summary for each of the papers.

2 On DNA, proteins, and gene activities

There is no exact definition of what a *gene* is as it is used slightly differently depending on the context. Historically, its definition has evolved from the time pre-dating the discovery of DNA (deoxyribonucleic acid) when a gene was just "the atom" of an inheritable trait (now a genotype) and which later was pinned down to be a region on a chromosome. Now a gene is said to be a specific region on the DNA that produces a functional product, that is a protein (or a special type of RNA (ribonucleic acid) molecule). Proteins are complex macro molecules that are essential for the function, structure, and dynamics of a cell. Each protein has a unique function. A cell interacts and responds to its environment by means of proteins and other molecules through a complex regulatory network. By studying these regulatory networks, scientists aim to understand and develop better treatments for complex diseases such as cancer, brain disorders, diabetes, but also viral diseases such as Acquired Immune Deficiency Syndrome (AIDS) and Severe Acute Respiratory Syndrome (SARS).

A gene is said to be activated or *expressed* when it, or more precisely, its coded information, is converted into proteins. Note, there is no single definition for this term. Somewhat simplified, gene expression takes place in two steps. In the first step, the gene's so called *template strand* of the double-helix DNA is transcribed into messenger RNA (mRNA), which is a complementary copy of the sequence of nucleotides (bases) on the DNA. In the second step, the mRNA moves out of the nucleus into the cell's cytoplasm and gets *translated* into a protein. This is done such that continuous triplets of mRNA nucleotides (codons) are converted into a sequence of amino acids that folds into a protein. Because the same amino-acid sequence can be subsequently modified in different ways, but also for other reasons, there is not a one-to-one relationship between the DNA and the final functional proteins; there are many more types of proteins than genes. Moreover, several protein copies of the same gene can be produced, which is done by the transcription of multiple mRNA copies or by multiple translations of the same mRNA copy. The more product, the more the gene is expressed. In biology, the aforementioned process is referred to as the *Central Dogma of Biology*. For further details see [1, 9].

When a cell responds to its environment, undergoes division, is attacked by viruses etc., different genes are activated and expressed in various amounts. By studying the gene-expression levels of cells under various (controlled or uncontrolled) conditions, researchers wish to infer the properties and regulatory networks of the cells. For various reasons not discussed here, it is easier to study the gene expressions by measuring the amount of mRNA than the amount of proteins. Standard protocols have been developed making it possible for any laboratory to extract mRNA from a cell line or a tissue of interest. Then, using one of a few techniques that separate (sort) the large number of mRNA sequences and measuring their individual copy numbers, estimates of the gene-expression levels can be obtained. One such recent method is the *microarray technology*.

3 The revolutionizing technology

The microarray technology has revolutionized the research area on *gene-expression analysis* by providing a relatively cheap and high-throughput platform for quantifying tens of thousands of gene expressions simultaneously. This makes it possible to collect huge sets of data that can be analyzed in order to infer functions of the cells. These data sets of high dimensions raise many new and interesting questions on statistical methodology. Some of them may be answered using classical theory of multivariate and cluster analysis, Bayesian statistics, robust and nonparametric statistics, variance components techniques and multiple testing, but others require new tools from these (and other) fields; see [17]. This calls for more statisticians and mathematicians to get involved.

The most general way to think of a microarray-based gene-expression experiment is as a black box. In one end the cell sample of interest is inserted and in the other end its gene-expression levels are output. However, as this thesis shows, in order to get correct and high quality measurements this box has to be opened and its components need to be investigated.

3.1 Brief about the technology

A microarray has well defined regions (*probes*) that each consists of immobilized sequences of DNA that are unique to a specific gene. When fluorophore labeled complementary DNA (cDNA) sequences (*targets*), obtained by reverse transcription of mRNA extracted from the samples of interest are *hybridized* to the probes, each region on the microarray will specifically bind a certain amount of DNA unique to the corresponding gene. Depending on if a two-channel or single-channel microarray platform is used, either several and differentially labeled targets are hybridized to the same array, or different targets are each hybridized to an exclusive array using identical labels. Fluorescent and radioactive molecules are standard types of labels. Next, the array is scanned at different wavelengths to excite the fluorescent molecules using a light source, typically a laser. Shortly after being excited, the fluorophores emit photons, which are registered and quantified in each position resulting in one high-resolution digitized image for each channel. Using image analysis methods, the pixels that belong to the regions that contain the probes are identified and their intensities are averaged giving estimates of the transcript abundances for all genes involved. For visualization purposes, the red and the green images are often combined into one image where the spots (probes) are falsely colored on a scale from red to yellow to green corresponding to the amount of relative gene expression. A schematic overview of this process is given in Figure 1. Spotted DNA microarrays [15] is a multi-channel technique that

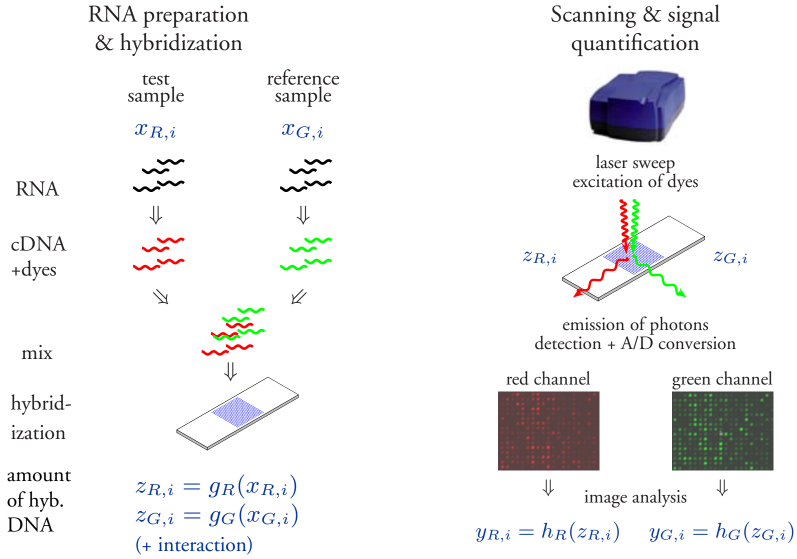


Figure 1: A schematic overview of a two-channel microarray technology. The gene-expression level, that is, the amount of mRNA, for gene i in the red and the green channel is $x_{R,i}$ and $x_{G,i}$, respectively. For each sample, the mRNA is extracted, purified, reversely transcribed into cDNA which is labeled with a dye that is unique for each channel. The labeled cDNA is mixed in equal amounts and hybridized to the probes on the microarray glass slide. The amount of hybridized cDNA of gene i is denoted by $z_{R,i}$ and $z_{G,i}$, respectively. To be explained in details below, we formally denote these steps by the submeasurement functions g_R and g_G. Next, with help of a scanner and image-analysis procedures, we try to estimate the amount of hybridized cDNA. The estimates are called y_R and y_G and the latter steps are denoted by the functions h_R and h_G, respectively.

measures the (relative) abundances of mRNA of 5,000-50,000 genes from two (or more) samples by competitive co-hybridization of aliquot labeled cDNA targets. To date, two-channel microarrays are by far the most commonly used types of arrays. Contrary, microarray techniques such as nylon membrane microarrays and high-density (100,000-500,000 probes) oligonucleotide in-situ microarrays [8] measure the abundances of gene expressions in one single sample. For a further introduction to the microarray technology, see Papers B, E, and D in this thesis. For a thorough description and history of the microarray technology, see [14].

3.2 Other usage

The DNA microarray technology is not limited to measure gene-expression levels and to pinpoint genes that are characteristic to a certain cell type, although most frequently used so. To give a few examples, the technique is also used to sequence DNA and detect genomic abnormalities such as amplifications and deletions of DNA. Low-resolution classical comparative genomic hybridization (CGH) methods, which are widely used in cancer research, are being replaced by more high-throughput microarray-based alternatives. Furthermore, with the advent of microarrays it has become practically and economically feasible to provide diagnostic tests on an individual basis as a complement to traditional clinical tests. More targeted treatments with less suffering for the patients can be developed.

3.3 Calls for improvements

There is a continuous and exponentially growing stream of publications with biological results of great value that are obtained by means of microarrays. Although the technology is highly praised and many advances have been made since the first proof-of-concept experiments performed a decade ago [15], there is also an increasing number of calls for improvements of protocols and statistical methods [7]. One crucial problem is that data generated with equal biological setups but by different laboratories and/or by different platforms may not easily be combined and integrated into the same analysis. This is even the case when experiments generated over time by the same facility are compared. To give a real-world example, a local collaborator is facing the tremendous task to pre-process and analyze data from almost 1000 hybridizations of scarce RNA extracts. These experiments are produced over a period of more than one year including many different cell cultures, different reagents, different people, various print batches (including different array types), climate conditions and so on. The first major task is to identify and remove systematic variation that obscures the biological signals by a process referred to as *normalization and calibration* or, more general, as *low-level analysis*. This thesis provides substantially improved methods for calibration and normalization of microarray data.

4 On microarray and gene-expression analysis

To give a more detailed motivation, but also a better understanding for the studies presented in this thesis, we will next give *our* view of microarray experiments from the perspective of gene-expression analysis. We will allow ourselves to generalize as much as possible in order to keep the discussion simple and we ask the reader to keep in mind that exceptions may exist.

4.1 Gene expression

In the context of microarray analysis, we refer to *number of gene transcripts*, that is, the number of *mRNA transcripts* for a specific gene available at a certain time, as the *expression level* of that gene. Note that this is a refinement of the more general "definition" introduced in Section 2. Although the sensitivity and specificity of the microarray techniques improve continuously, gene-expression levels are still not measured on a single-cell basis. In reality, the average expression level in a large number of cells is measured. Thus, when talking about the expression level of a gene in the context of microarray experiments it is typically meant the *average number of mRNA transcripts* in a set of cells. To formalize this, assume we are interested in a set of genes $\mathcal{I} = \{1, \ldots, I\}$ in the cell samples $\mathcal{C} = \{1, \ldots, C\}$ where I is the number of genes and C is the number of cell samples. Let N_c, which is typically large, be the total number of cells in sample $c \in \mathcal{C}$, and let $n_{c,i}$ be the total number of gene transcripts for gene $i \in \mathcal{I}$ in all those cells. Then, the average number of gene transcripts per cell for gene i is $x_{c,i} = n_{c,i}/N_c$. We refer to $x_{c,i}$ as the *gene-expression level* for gene i in sample c.

4.2 Expressed and differentially expressed genes

The simplest microarray experiment has only one cell sample, that is $C = 1$. Ideally, it allows us to classify genes to either be *non-expressed* or *expressed*. For a gene to be non-expressed no gene transcripts are allowed, otherwise it is expressed. Formally, for sample c, genes $\mathcal{I}_{x_c=0} = \{i \in \mathcal{I} : x_{c,i} = 0\}$ are labeled "non-expressed", and all other genes, $\mathcal{I}_{x_c \neq 0} = \mathcal{I} \backslash \mathcal{I}_{x_c=0}$, are labeled "expressed". However, rather than being either on or off in this sense, in reality a gene can be expressed with any number of gene transcripts. Moreover, it does not make much sense to describe the gene activity as the (absolute or average) number of gene transcripts. For instance, for a certain gene it might be enough to be expressed in one mRNA copy in order to affect a cell's state whereas for another gene there might be a threshold of, say, 2000 transcripts in order to make a difference. It can also be the opposite that a gene is normally expressed in 2000 transcripts per cell and makes a difference first when its expression level drops below 500 transcripts, and so on. Instead, it is better to quantify the amount of activity as the expression levels *compared to a reference*. The reference should not be the level of another gene, because it makes little sense to compare $x_{c,i}$ with $x_{c,j}$ for $i \neq j$, but instead the expression level of the same gene under other circumstances, that is $x_{c,i}$ compared to $x_{d,i}$ for $c \neq d$[1]. As an example, it is more informative to compare the expression level of a promoter oncogene in a tumor tissue to its expression level in a healthy tissue, than to compare the expression level of a promoter oncogene with the expression level of an inhibitor oncogene. Thus, in addition to classifying genes as non-expressed or expressed, an *experimental setup with two or more samples* allows us to classify genes as *differentially* or *non-differentially expressed*. Formally the set of genes labeled "non-differentially expressed" is $\mathcal{I}_{x_c=x_d} = \{i \in \mathcal{I} : x_{c,i} = x_{d,i}\} = \{i \in \mathcal{I} : N_d n_{c,i} = N_c n_{d,i}\}$ for some $c \neq d$ with $c, d \in \mathcal{C}$. All other genes, $\mathcal{I}_{x_c \neq x_d} = \mathcal{I} \backslash \mathcal{I}_{x_c=x_d}$, are labeled "differentially expressed".

Given the above foundation and binary classification schemes, it is possible to test rather complicated biological hypotheses of different flavors involving the cell samples of interest. By far, the most common application of microarray experiments to date has been to scan for genes that are differentially expressed, that is, to test the null hypothesis H_0 against H_1;

$$\begin{aligned} H_0 &: x_{c,i}{=}x_{d,i} \\ H_1 &: x_{c,i}{\neq}x_{d,i} \end{aligned} \tag{1}$$

[1] In some analyses such as cell-cycle studies, it may indeed be of interest to look at the expression level of a single gene in relation to all genes, that is, $x_{c,i}/\sum_{j\in\mathcal{I}} x_{c,j}$.

for all genes $i \in \mathcal{I}$ and in samples $c \neq d; c, d \in \mathcal{C}$.

4.3 Relative gene expression and ordering of expression levels

In addition to the above, many experiments try to order the gene-expression levels relative to each other. For instance, the question asked may be if the expression of gene i is increasing between time t_1 and t_0 or not. This can be done by testing the null hypothesis $\tilde{H}_0$ against $\tilde{H}_1$;

$$\begin{aligned} \tilde{H}_0 &: x_{c,i} > x_{d,i} \\ \tilde{H}_1 &: x_{c,i} \leq x_{d,i} \end{aligned} \tag{2}$$

where c and d represent the samples obtained at time t_1 and t_0, respectively. Hence, this test provides a way to *order* gene-expression measurements of the same gene. We say that gene i is *up-regulated* in sample c compared to sample d if $\tilde{H}_0$ is true, and vice versa if it is *down-regulated*.

Continuing, when comparing gene-expression levels of two samples, typically the *relative gene-expression* in one sample *compared to* another[2] is considered, that is $x_{c,i}/x_{d,i}$ for $c \neq d$ where $c, d \in \mathcal{C}$ and $i \in \mathcal{I}$. Since $x_{c,i}/x_{d,i}$ is not symmetric in $x_{c,i}$ and $x_{d,i}$, it is more common to take the logarithm because its absolute value is the same regardless of whether gene i is up- or down-regulated by a certain fraction. Taking the logarithm is so common that the transform has its own name and its own symbols; the *log-ratio* and the (average) *log-intensity* for gene i [20] is

$$\begin{aligned} M_i &= \log_2 \frac{x_{c,i}}{x_{d,i}} = \log_2 x_{c,i} - \log_2 x_{d,i} \\ A_i &= \frac{1}{2} \log_2(x_{c,i} x_{d,i}) = \frac{\log_2 x_{c,i} + \log_2 x_{d,i}}{2}, \end{aligned} \tag{3}$$

where, due to the binary nature of the image-analysis quantified signals, base two has become a de facto standard, although any base would do. M and A are mnemonics for “minus” and “add”, respectively [16]. *Estimates* of the gene-expression levels may be non-positive. For this reason, the second step of equalities are included to make it explicit that in such cases the log-ratios and the

[2] We here allow ourselves to think of the channels in two- or multi-channel microarray experiments conceptually as separate measurements performed in parallel.

log-intensities are *not* defined. Moreover, as long as the signals are positive the transform is bijective. It follows that, for up-regulated genes $M_i > 0$ and for down-regulated $M_i < 0$. Non-differentially expressed genes have $M_i = 0$. That is, under transform (3) the null hypothesis H_0 and the alternative hypothesis H_1 can be written as

$$\begin{aligned} H_0 &: M_i{=}0 \\ H_1 &: M_i{\neq}0 \end{aligned} \tag{4}$$

for all genes $i \in \mathcal{I}$ where it is implicit that samples $c{\neq}d; c, d \in \mathcal{C}$ are considered, and analogously for hypothesis (2).

To interpret the value of $x_{c,i}/x_{d,i}$ beyond the ordering of expression levels, additional biological assumptions have to be added. For instance, it is reasonable to assume that when the gene expression changes, the gene effect (phenotype) changes too, and that an additional small change in the same direction *ceteris paribus* will further change the gene effect in the same direction. We may also assume a *linear functional relationship* between gene-expression level and gene effect locally. On the other hand, it is much harder to assume a linear relationship on the global scale. This is also unlikely to be true. Even worse, the biological effect of one gene is likely to be affected by the biological output of other genes and so on. Considering the huge number of genes, this illustrates the essence of the enormous task researchers in the field of *functional genomics* are facing trying to infer genetic regulatory networks of the cells based on gene-expression data.

4.4 Fundamental assumptions and approximations

A gene-expression experiment based on, say, microarrays, ideally gives estimates that are proportional to $\{n_{c,i}\}_{c,i}$, but they include no estimates of $\{N_c\}_c$ per se. However, in order to classify genes as differentially or non-differentially expressed according to (1) or (4), we have to know the *relative number of cells* in the two samples compared, that is, N_c/N_d for $c, d \in \mathcal{C}$. In practice, this is done by assuming that N_c is approximately proportional to the sum $\sum_{i\in\mathcal{I}} n_{c,i}$, which implies that it is assumed that all mRNA transcripts have equal (physical) weight and that at any time there is a large set of genes that are expressed. More precisely, this is typically assumed implicitly in the step of the microarray protocol where it is said that aliquot extracted and purified total amount of mRNA (or labeled cDNA) should be used. Any microarray-based comparative gene-expression analysis relies on these types of fundamental assumptions. Without them there would be no reference available[3]. In this thesis we will not discuss these assumptions and approximations further but assume them to be correct or at least reasonable enough.

4.5 Measurement functions

In reality, microarray experiments are much more complicated than outlined in Section 3.1, which has immediate implications on the data analysis. To summarize the model notation used in Papers D and E, let the complete microarray process from gene-expression levels $\{x_{c,i}\}_{i\in\mathcal{I}}$ to the corresponding set of quantified signals $\{y_{c,i}\}_{i\in\mathcal{I}}$ in channel c be represented by the unknown *measurement function* f_c such that in the error-free case we have

$$y_{c,i} = f_c(x_{c,i});\ \forall c, i. \tag{5}$$

To avoid complicating the model too much, we assume that f_c is the same for all genes. However, it is straightforward to extend the definition such that the measurement function is specific to disjoint subgroups of genes such as print-tip groups, plate groups, tissue-specific gene groups etc. A plate group is a set of spots that originate from the same microtiter plate, and analogously, a print-tip group is a set of spots that were printed by the same print tip. More complicated, but nevertheless possible, is to define spatially dependent measurement functions. Note, we *do* assume that for any two channels $c{\neq}d$, the observed signals $y_{c,i}$ and $y_{d,i}$ are independent given the gene-expression levels $x_{c,i}$ and $x_{d,i}$. A generalized definition of a measurement function that does not assume separate components is $(y_{1,i},\ldots,y_{C,i}) = \mathbf{f}_{\mathcal{C}}(x_{1,i},\ldots,x_{C,i});\ \forall i$. Note that (5) would not be true for two-color microarrays if there were *spectral cross-talk* between channels, which may occur if for instance excitation and emission spectra of the two fluorophores overlap. Any cross-talk is assumed to be negligible or already corrected for. We

[3] Alternatively one could *define* the gene expression to be a gene's relative weight of mRNA transcripts compared to the total weight of all genes and in all cells, that is $x_{c,i} = w_{c,i}/\sum_{j\in\mathcal{I}} w_{c,j}$ where $w_{c,i}$ is the weight of all mRNA transcripts of type i in sample c. The weight $w_{c,i}$ of all sequences for gene i depends on the number of sequences $n_{c,i}$ and their lengths and compositions. This definition would relax some of the assumptions and approximations. However, it would at the same time move the definition of gene expression away from what we normally would mean by a gene being expressed.

will return to this assumption when introducing the concept of submeasurement functions below.

Avoiding quenching, scanner saturation, and other non-linear phenomena, we assume that f_c is strictly increasing. In such case, a unique inverse, f_c^{-1}, exists and $x_{c,i}$ can be calculated by the back transform (still assuming no errors)

$$x_{c,i} = f_c^{-1}(y_{c,i}). \tag{6}$$

4.5.1 Testing for differentially expressed genes

Returning to our test for differentially expressed genes and assuming zero noise, we see that when the measurement functions are identical, that is, when

$$f = f_c = f_d; \ c, d \in \mathcal{C}, \tag{7}$$

the set of non-differentially expressed genes can be expressed as $\mathcal{I}_{x_c=x_d} = \{i \in \mathcal{I} : x_{c,i} = x_{d,i}\} = \{i \in \mathcal{I} : f^{-1}(y_{c,i}) = f^{-1}(y_{d,i})\} = \{i \in \mathcal{I} : y_{c,i} = y_{d,i}\}$. If $f_c \neq f_d$, the last equality will not hold anymore.

4.5.2 Linear and affine measurement functions

A common assumption, implicit or explicit, is that the measurement function is *linear* (with zero intercept);

$$f_c(x_{c,i}) = b_c x_{c,i} \iff f_c^{-1}(y_{c,i}) = \frac{y_{c,i}}{b_c} \tag{8}$$

for channel c and gene i where $b_c \neq 0$. This measurement function simplifies downstream statistical analysis and allows direct interpretation of the observed signals, that is, the signals are assumed to be *proportional* to the corresponding gene-expression levels. However, it does not capture systematic effects that express themselves as channel biases, which for instance is the case with imperfect background correction, scanner biases, etc. The *observed* log-ratios and log-intensities under a linear transform are

$$\begin{aligned} M_i &= \log_2 \frac{f_c(x_{c,i})}{f_d(x_{d,i})} = \log_2 \beta + \log_2 \frac{x_{c,i}}{x_{d,i}} \\ A_i &= \frac{1}{2}\log_2(f_c(x_{c,i}) f_d(x_{d,i})) = \frac{1}{2}\log_2(b_c b_d) + \frac{1}{2}\log_2(x_{c,i} x_{d,i}) \end{aligned} \tag{9}$$

where $\beta = b_c/b_d$ and $b_c, b_d \neq 0$. Thus, the linear transform introduces a bias $\log_2(\beta)$ in the log-ratios, which is seen as a (constant) vertical shift in the log-ratio versus log-intensity plots. See Figure 2 for an example. There is also a (constant) shift in the log-intensities, which is of less importance since the log-ratios are the quantities of interest.

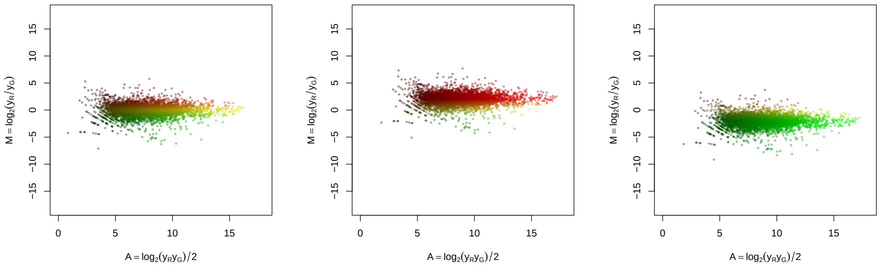


Figure 2: Example of log-ratio versus log-intensity scatter plots between channels $\{R, G\}$ where a linear transform introduces a constant bias in the observed log-ratios. The same data set is used in all cases. *Left:* There is a balanced scale in the channels, that is, $\beta = b_R/b_G = 1$. *Middle:* Relative scale is $\beta = 4/1$, which introduces a $log_2\beta = +2$ shift in the log-ratios. *Right:* A relative scale $\beta = 1/4$ introduces a $log_2\beta = -2$ shift. The data points are colored on a red-to-yellow-to-green scale corresponding to the observed log-ratios and a dark-to-bright scale corresponding to the observed log-intensities to imitate the (false) colors of the spots as seen in typically two-color microarray images.

The second simplest measurement function to consider, which is also more powerful, is the *affine* measurement function

$$f_c(x_{c,i}) = a_c + b_c x_{c,i} \iff f_c^{-1}(y_{c,i}) = \frac{y_{c,i} - a_c}{b_c} \qquad (10)$$

where $b_c \neq 0$. The affine transformation can be motivated physically, biologically, and mathematically; see Papers D and E. To continue, the *observed* log-ratios under an affine transform are

$$
\begin{aligned}
M_i &= \log_2 \frac{f_c(x_{c,i})}{f_d(x_{d,i})} \\
&= \log_2 \beta + \log_2 \frac{\frac{\alpha_i}{2} + \sqrt{\frac{\alpha_i^2}{4} + \frac{x_{c,i}}{x_{d,i}} \beta 2^{2A_i}}}{-\frac{\alpha_i}{2} + \sqrt{\frac{\alpha_i^2}{4} + \frac{x_{c,i}}{x_{d,i}} \beta 2^{2A_i}}} + \log_2 \frac{x_{c,i}}{x_{d,i}} \qquad (11)
\end{aligned}
$$

where $\alpha_i = a_c - \beta a_d(x_{c,i}/x_{d,i})$ and A_i is the observed log-intensity for gene i. In addition to the constant bias seen in the linear transform there is a bias in the log-ratios that depends on the size of true relative gene-expression ratio $x_{c,i}/x_{d,i}$ and the (observed) log-intensities A_i, cf. Figure 3. See Paper D for details. The existence of this *non-linear intensity-dependent bias* complicates test (4) for non-differential expression. Of less importance, there is also an intensity and relative-gene-expression dependent bias in the observed log-intensities.

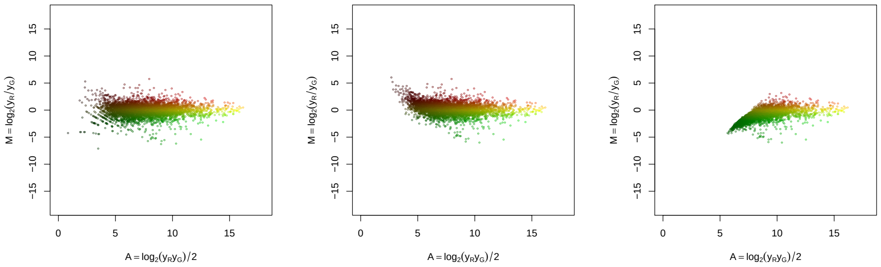


Figure 3: Example where an affine transform introduces an intensity-dependent bias in the observed log-ratios. The same data set with relative scale $\beta = 1/1$ is used in all cases. *Left:* No bias in either channel. *Middle:* A bias of size 20 in channel R, that is $(a_G, a_R) = (0, 20)$. *Right:* An additional bias of size 200 in the other channel so that $(a_G, a_R) = (200, 20)$.

4.5.3 Submeasurement functions

The concept of a measurement function can be applied to any part of the microarray process, not only the complete process from gene expression to the quantified signals in the computer. For instance, as done in Paper E, we can choose to focus on the *submeasurement function* describing the scanner and the image-analysis steps. This is also illustrated in Figure 1 where the complete measurement function f_c is split up into two subsequent submeasurement functions g_c and h_c, that is, $f_c = h_c \circ g_c$ assuming the two channels $\mathcal{C} = \{R, G\}$ act separately. Submeasurement functions can be defined and studied at any resolution. Furthermore, when investigating single submeasurement functions alone, the assumption (5) that the measurement functions are separate between channels may be less questionable or at least has less impact on the final conclusions. Notice that linear and affine submeasurement functions h_c and g_c are *closed under composition*, which implies that f_c belongs to the same class. This is very convenient, not only in theory but also in practice, because it allows some normalization methods to be applied in any order, which makes it possible to postpone some required decisions to a later step in the analysis process.

4.6 Calibration and normalization

Systematic effects or *artifacts* in the observed signals are observed when the measurement functions are non-linear. As exemplified in Section 4.5.2 and illustrated in Figure 3, *calibration* and *normalization* is about identifying and removing such unwanted *bias* in the observed signals in order to reveal the true relative gene-expression levels.

4.6.1 Calibration

When a parametric model for f_c is used and data is of the form $\{(x_{c,i}, y_{c,i})\}_i$, the measurement function f_c can be computed in the noise-free case. The true expression levels are then recovered through

$$\hat{x}_{c,i} = \hat{f}_c^{-1}(y_{c,i}); \forall c, i, \tag{12}$$

where $\hat{f}_c$ is the computed version of f_c. We refer to a method that calculates the *calibrated signal* $\hat{x}_{c,i}$ by (12) as a *calibration method*[4]. An example is when known amounts of certain RNA, also known as *spike-ins*, for a set of genes $\mathcal{I}_{\text{known}}$ are added to the original set of RNA transcripts, then measured and observed in

[4] Without noise, we often have $\hat{f}_c = f_c$. However, we also regard the case when f_c is known only up to a multiplicative constant ($\hat{f}_c \propto f_c$) as a calibration method. In that case $\hat{x}_{c,i} \propto f_c^{-1}(y_{c,i}) = x_{c,i}$. This is because the absolute scale of $x_{c,i}$ is of subordinary interest.

order to obtain $\{(x_{c,i}, y_{c,i})\}$ for $i \in \mathcal{I}_{\text{known}}$. With classical *regression techniques*, the measurement function $\tilde{f}_c$ for the spike-ins can be estimated by $\hat{\tilde{f}}_c$. With the assumption that the other genes have the same measurement function, that is $f_c = \tilde{f}_c$, we have $\hat{f}_c = \hat{\tilde{f}}_c$ and all $x_{c,i}$ can be estimated from (12). For example, with known expression levels and observations of the spike-ins, and assuming an affine transformation, the bias and scale parameters a_c and b_c in channel c can be estimated, respectively. Assuming the same parameters for the measurement function of all other genes, estimates of the gene-expression levels can be obtained.

In Paper E, another type of calibration method is discussed. It does not estimate the complete measurement function, but instead a *submeasurement function* specific to the scan and image analysis steps of the microarray process. For the analysis of that paper, there are no known data points $\{x_{c,i}\}$ available, but the data points are known to be *fixed* between repeated measurements at different sensitivity levels. We impose an additional assumption about the measurement function stating that no matter what sensitivity level $k = 1, \ldots, K (K \geq 2)$ is used the bias is the same;

$$y_c^{(k)} = f_c^{(k)}(x_{c,i}) = a_c^{(k)} + b_c^{(k)} x_{c,i} = a_c + b_c^{(k)} x_{c,i}, \; \forall c, i, k. \tag{13}$$

Here superscript (k) denotes sensitivity level with index k. The sensitivity is adjusted by modifying the photomultiplier tube (PMT) gain. With this additional constraint, which can be motivated physically and is strongly supported by data, it is possible to infer the submeasurement function based solely on the observations $y_c^{(k)}$ (up to a scale factor). Thus, in the absence of noise, the signals $x_{c,i}$ of interest can be estimated up to a multiplicative constant, which is the reason why this method is a calibration method.

4.6.2 Normalization

Even if no calibration data points such as spike-ins or fixed data points are available, it is still possible to infer something about the measurement functions. Consider a two-sample experimental setup for which it is possible to identify a *subset* of non-differentially expressed genes, $\mathcal{I}_{\text{norm}} = \{i \in \mathcal{I} : x_{c,i} = x_{d,i}\} \subset \mathcal{I}_{x_c = x_d}$ for $c \neq d$. This is typically done by assuming that *most genes are non-differentially expressed*. With this set we can find a smooth *normalization function* such that $g_c^{-1}(y_{c,i}) = g_d^{-1}(y_{c,i})$ for $\mathcal{I}_{\text{norm}}$. This is analogue to the above where we used separate calibration functions for each channel. Next, assume that this is true for *all* non-differentially-expressed genes, and otherwise not, that is,

$$x_{c,i} = x_{d,i} \Longleftrightarrow g_c^{-1}(y_{c,i}) = g_d^{-1}(y_{d,i}) \tag{14}$$

for $i \in \mathcal{I}$. A *normalization method* is then a method that transforms the signals as

$$\tilde{y}_{c,i} \leftarrow g_c^{-1}(y_{c,i}); \forall c, i. \tag{15}$$

When the channels are normalized separately, orthogonality between channels will be preserved, otherwise artificial *cross-talk* will be introduced. We immediately note that g_c^{-1} can be chosen as the inverse of the measurement functions f_c^{-1}, if known. In other words, calibration is a special case of normalization. Moreover, note that it is always possible to choose, say, g_c as the identity function such that $x_{c,i} = x_{d,i} \Longleftrightarrow y_{c,i} = g_d^{-1}(y_{d,i})$ for $i \in \mathcal{I}$. As an example, consider the affine model (10) with two channels c and d, and suppose there are no fixed or spike-in data points available. Then a_c, b_c, a_d and b_d cannot be estimated. However, without noise we have

$$y_{c,i} = \alpha_i + \beta \frac{x_{c,i}}{x_{d,i}} y_{d,i} \tag{16}$$

with α_i and β as in (11). Hence, if there is a known set $\mathcal{I}_{\text{norm}}$ of non-differentially expressed genes, the corresponding points $\{(y_{c,i}, y_{d,i})\}_i$ are located along a line with slope β and intercept $\alpha = a_c - \beta a_d$. Therefore, the parameters α and β are known (or can be estimated when noise is present). Hence, the normalized signals

$$\begin{aligned} \tilde{y}_{c,i} &= y_{c,i} = a_c + b_c x_{c,i} \\ \tilde{y}_{d,i} &= \alpha + \beta y_{d,i} = a_c + b_c x_{d,i} \end{aligned} \tag{17}$$

are computable and satisfy (14). Moreover, it follows from (11) that the intensity-dependent bias is eliminated ($\alpha_i = 0$) for the log-ratios $\tilde{M}_i = \log_2(\tilde{y}_{c,i}/\tilde{y}_{d,i})$ of all *non-differentially* expressed genes.

Another example of normalization is given in Paper E (and D). There we show that for some scanners there exist PMT-independent (scale-independent) biases $a_c^{(k)} = a_c$ as in (13). Thus, by adjusting the gain of the two PMT and hence the scale parameters b_c and b_d, it is possible to find a configuration such that

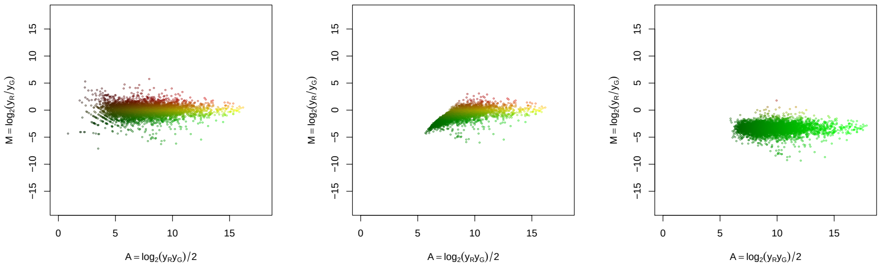


Figure 4: Example where an adjustment of the gain (scale parameters) can normalize affinely-transformed signals, that is, remove the curvature. The same data set is used in all cases. *Left:* No bias and equal scale in both channels. *Middle:* The same data with channel biases $(a_G, a_R) = (200, 20)$ and equal scale $(\beta = 1)$. *Right:* The same biases now compensated by the relative scale $\beta = 1/10$. A constant bias in the log-ratios remains, but there is no curvature for the non-differentially expressed genes.

$\alpha_i = 0$ for $i \in \mathcal{I}_{x_c = x_d}$ in Equation (11) resulting in no curvature for the non-differentiated genes. Most likely this is done in quite a few laboratories when the arrays are pre-scanned and the PMT settings are adjusted such that the two channels are "balanced". This rather mechanical normalization procedure is illustrated in Figure 4. Note, if the PMT-independent bias would be zero, adjusting the PMT gains would just shift the log-ratios the same amount up or down for all intensities as in Figure 2. Having said this, a quick and simple test for linearity of a scanner's submeasurement function is then also given.

It follows immediately from (14) that normalized signals can be used to test for non-differential expression; testing H_0 against H_1 in (1) is the same as testing $H'_0 : \tilde{y}_{c,i} = \tilde{y}_{d,i}$ against $H'_1 : \tilde{y}_{c,i} \neq \tilde{y}_{d,i}$ for $c \neq d$. However, testing $\tilde{H}_0$ against $\tilde{H}_1$ in (2) is not necessarily the same as testing $\tilde{H}'_0 : \tilde{y}_{c,i} > \tilde{y}_{d,i}$ against $\tilde{H}'_1 : \tilde{y}_{c,i} \leq \tilde{y}_{d,i}$ for $c \neq d$. Although the latter is less likely to be a problem in reality, more problematic is the ordering of *relative* gene-expression levels. Observing $\tilde{y}_{c,i_1}/\tilde{y}_{d,i_1} > \tilde{y}_{c,i_2}/\tilde{y}_{d,i_2}$ for $i_1, i_2 \in \mathcal{I}$ does not imply $x_{c,i_1}/x_{d,i_1} > x_{c,i_2}/x_{d,i_2}$. To illustrate this with a real-world example, consider a two-channel microarray experiment with replicated spots i_1 and i_2 of the same gene i. Although the underlying relative expression levels for the two spots are identical by definition,

$x_{c,i}/x_{d,i}$, due to a non-linear measurement function and differences in spot concentrations, the normalized log-ratio can be greater for spot i_1 than for spot i_2, which becomes a problem when (if) averaging log-ratios. This is why we differentiate between *calibration* and *normalization*.

Other normalization methods such as quantile normalization methods and curve-fit normalization methods can be seen in the light of (14) and (15). For instance, a curve-fit (intensity-dependent) normalization method normalizes the signals by assuming $M_i = h(A_i)$ for all non-differentially expressed genes. The normalization method in this case differs slightly from (15) and acts jointly on the two channels c and d through the log-ratio. The normalized log-ratios are $\tilde{M}_i = M_i - h(A_i)$, where h is estimated using a robustified version of some nonparametric smoothing method such as local polynomial regression (for instance loess or smoothing splines). Putting $\tilde{y}_{c,i} = 2^{-0.5h(A_i)}y_{c,i}$ and $\tilde{y}_{d,i} = 2^{0.5h(A_i)}y_{d,i}$, we find that $x_{c,i} = x_{d,i}$ is equivalent to $\tilde{y}_{c,i} = \tilde{y}_{d,i}$ for $i \in \mathcal{I}_{\text{norm}}$.

4.7 Error models and error propagation

Error terms are still to be discussed. Any measurement contains noise

$$y_{c,i} = f_c(x_{c,i}) + \varepsilon_{c,i} \tag{18}$$

where $\{\varepsilon_{c,i}\}_{c,i}$ are independent and unobserved zero-mean error terms for $i \in \mathcal{I}$ and $c \in \mathcal{C}$. With noise, the estimate (12) of $x_{c,i}$ contains two sources of error: the possible estimation error of $\hat{f}_c$ (as an estimate of f_c) and the presence of $\varepsilon_{c,i}$ in (18). For this reason, $\hat{x}_{c,i}$ will usually depart from $x_{c,i}$ even when f_c is estimated with good accuracy. For normalization methods we may still use (15) provided g_c^{-1} is replaced by an estimate $\hat{g}_c^{-1}$. However, formula (14) is only valid in the noise-free case although it still serves as a guiding principle for defining the normalization method.

With error terms, the test of H_0 against H_1 becomes a *statistical test*; the region were H_0 is rejected is a region defined by the distribution of the error terms.
Various error models have been suggested. The most naïve one assumes independent and identical distributed (i.i.d.) normal error terms for all genes, say, $\varepsilon_{c,i} \in N(0, \sigma_c^2); \forall c$. The exact error model depends on the which (sub)measurement function is studied, but typically a heteroscedastic model is more useful, that is,

a model where the standard deviation of the noise is a function of the signal. An example is

$$\sigma_{c,i}(x_{c,i}) = \delta_c + \lambda_c x_{c,i} \tag{19}$$

where $\delta_c \geq 0$ and $\lambda_c \geq 0$. Another error model with additive and multiplicative noise was suggested by [11, 12] and utilized in [4, 5]. The latter two suggest variance stabilizing models that also contain an affine component. Depending on measurement function and noise structure, the errors propagate differently resulting in a more or less complicated noise structure of, say, the log-ratios, which in turn affects the region of rejection in a log-ratio-based hypothesis test. This thesis does not explicitly study error propagation further, but Paper G suggests a heteroscedastic affine error model for microarray data with dilution series that allows us to identify a calibration function. See also Section 6.7, which gives a brief overview of this paper. More research on error models for microarrays is needed.

4.8 The search for an omnibus way to compare calibration and normalization methods

Quite a few normalization methods and some calibration methods for single-channel and two-channel microarray data have been published to date. For a description of some of them, see Paper D. Depending on the error structure, a test of H_0 against H_1 in (1) based on normalized signals may be as simple (or as hard) to conduct as a test based on calibrated signals. However, it is our belief that in general a correct calibration method is to prefer over a normalization method, because the former tries to estimate the true gene-expression levels (up to a scale factor) contrary to the latter. Not to be forgotten, calibration allows us to test $\tilde{H}_0$ against $\tilde{H}_1$ in (2). More important, though, is how to know which calibration or normalization method to use. This calls for quantitative methods in order to compare any two normalization or calibration procedures. Unfortunately, no generic method has been presented to date. It is still an open problem which calibration or normalization method is optimal for a given data set.

A reasonable criterion for a normalization method to be good is that normalized signals from repeated measurements of identical samples should be “as similar as possible”. Statistically we say that the *variability* (or variance) of the normalized signals should be small (or equivalent, that the *precision* should be high). In addition to this, *small bias* (high accuracy; "closeness to the truth") of the estimated signals is preferred, although not required by all tests. For example, consider a single two-channel microarray where each gene has been replicated K times for which we obtain log-ratios $\mathbf{M}_i = (M_{i,1}, \ldots, M_{i,K})$ for each gene i. Let $\mathbf{M} = \{\mathbf{M}_i\}_i$ be the set of all log-ratios. The average sample variance of all genes is

$$\bar{\sigma}^2(\mathbf{M}) = \frac{1}{I} \sum_{i=1}^{I} \hat{\sigma}_i^2(\mathbf{M}_i) \tag{20}$$

where $\hat{\sigma}_i^2(\mathbf{M}_i)$ is the sample variance of the log-ratios for gene i. Let this be our precision measure. Next, assume that we apply two different normalization methods, called A and B, to the same data set. This will result in two sets of normalized log-ratios, $\tilde{\mathbf{M}}^A = \{\tilde{\mathbf{M}}_i^A\}_i$ and $\tilde{\mathbf{M}}^B = \{\tilde{\mathbf{M}}_i^B\}_i$. By comparing $\bar{\sigma}^2(\tilde{\mathbf{M}}^A)$ and $\bar{\sigma}^2(\tilde{\mathbf{M}}^B)$ we then wish to say which of the two methods A and B that performs best on the given data set. However, we cannot use (20) alone for this purpose. One reason may be heteroscedasticity of the log-ratios. Another is that (20) is not invariant under affine transformations. To illustrate this, consider the case where method A is a "perfect" calibration method in the sense that there are no systematic effects remaining, and where method B is such that is first does what method A does, but then adds a large constant to both channels. The absolute value of the log-ratios for the latter method will decrease as the added channel bias gets larger resulting in $\bar{\sigma}^2(\tilde{\mathbf{M}}^B) < \bar{\sigma}^2(\tilde{\mathbf{M}}^A)$, although method A is the best by definition. A statistician would of course immediately object and remind that we do not control for bias. The accuracy of $\tilde{\mathbf{M}}^B$ will of course decrease as it departs from the true values. However, the problem in microarray analysis is that the truth is almost never known and therefore we have to rely on other methods. We have chosen the average standard deviation as an example because we know it is indeed used for the purpose of comparing normalization and calibration methods. In Paper B, we used a similar measure ourselves. Similar concerns occur when trying to define a distance space for clustering and classification.

A possible solution is to use a variance stabilizing transform such as [4, 5] before calculating (20). However, as these transforms are normalization methods themselves, it is not clear how to use them for the purpose of comparing signals normalized by other means. This is an open problem and more research is needed.

This thesis neither discuss nor provide answers to the above, but we hope that by better understanding measurement functions for microarrays we are one step closer to an answer. In the meantime, we have to rely on subjective judgements based on visual comparisons of log-ratio versus log-intensity plots, density plots, summary statistics, comparisons of "top 100 gene lists", comparisons to measurements from other techniques, and most importantly, biological validation of the results.

5 Making methods available to the research community

Since the end of year 2000, we have developed an open-source microarray analysis package in R [6] called *aroma* (formerly known as com.braju.sma) for the purpose of making existing and new statistical tools easily accessible to life science researchers. It is aimed to be user-friendly and is used by international research groups [19, 2] and by academia who in turn give us most valuable practical (tests and bug reports) and scientific feedback. In addition to this, aroma provides us with a structured and expandable platform to implement new statistical algorithms and compare them with published ones. The maintenance of this project has been and is still tightly coupled to computationally intensive projects of our own. Aroma pre-dates the promising Bioconductor.org project [18], which is one reason for why it is currently not available via that forum. We hope to be able to conform toward their standards and conventions in a way that is backward compatible with our standards [3] (and Paper C) and with users' existing aroma scripts. We plan to broaden the access to our statistical tools further by making the package available as a plug-in for the BioArray Software Environment (BASE) [13]. Part of this work has already been initiated by the group responsible for BASE. With the upcoming release of BASE v2 the opposite will also be true in which all of BASE's database and analysis tools can be accessed from within R and aroma via a Java application programming interface (API). The aroma package is presented in Paper F.

6 Summary of the papers

A brief presentation of the papers presented in this thesis follows. Naturally, our knowledge on microarray analysis has evolved in the course of our research, which is also reflected in the papers. The papers are listed in chronological order.

6.1 Paper A

Paper A was written as a complementary article to an online discussion at Science's Science Aging Knowledge Environment (SAGE KE). It was intended to give a short introduction of basic normalization methods and statistical tests for microarray data to an audience consisting of experimentalists. The one-sample t-test and the one-sample empirical Bayes' test ("B-test") [10] are explained and a simple empirical comparison of the two tests is performed illustrating how different methods and different number of arrays give different results. A simple comparison of the sets of identified genes from applying two different image analysis methods was also conducted. Data was provided by Matt Callow and Eric Rubin. Saira Mian contributed with the introduction and Terry Speed contributed with valuable scientific feedback. Brent Calder did the Axon GenePix image analysis and replicated the analysis done for the Spot image-analysis data, which was done by Henrik Bengtsson who also summarized the results and wrote the remaining text.

6.2 Paper B

Paper B is about print-order and plate effects in spotted microarray data. It introduces a novel way of displaying microarray data called *print-order plots*. When producing spotted microarrays the spots are not printed simultaneously, but subsequently in groups of size equal to the number print tips used by arrayer. The arrayer does a so called *source visit*, fills the print tips with the DNA clones in the microtiter-plate wells, and moves to the area where the glass arrays are mounted. After that, it will print a set of spots at once on the array, then move over to the second array and so on. The amount of DNA picked up at a single source visit will often last 50-100 arrays before a new source visit is needed. When the first round is completed another set of clones are picked up and printed on array one, two and so on until the arrays are filled. For this reason, it will take many hours to print the arrays, and, if the printing environment is not controlled well enough, the properties of the spots may vary over time. Thus, by displaying, say, the observed log-ratios in the order that the corresponding spots were printed, such abnormalities may be detected. The print-order plots illustrated in the paper show strong *apparent* print-order effects.

Different sequences of normalization methods applied to all genes or subset of genes are applied and the results are visually compared. In order to compare normalization strategies objectively, the paper suggest a *measure of reproducibility* (MOR), which is a precision measure of how *dissimilarly* replicated log-ratios are across arrays similar to how (20) is defined. Using this measure, one of the optimal strategies found was to apply a print-tip lowess normalization followed by a plate-group lowess normalization. Comparing the MOR before and after normalization, and comparing with print-tip lowess normalization alone, most of the systematic effects were removed by print-tip normalization (the MOR decreased from 100% to 46%) and succeeding plate-group lowess normalization would remove a little bit more (MOR decrease to 41%).

It turns out that the majority of the *print-order effects* seen in the data set studied are actually "fake" visual artifacts. They are due to the fact that different types of clones originating from different sets of microtiter plates were printed at different times. As shown in one of the figures in the paper, the brain clones were spotted during one day and liver clones during another day with the effect that the data points from the two groups are located in separate parts of print-order plots. Moreover, since the experiment was done with RNA extracted from liver, the genes corresponding to the liver clones were strongly expressed whereas the brain clones were hardly expressed at all. Together with the fact that there was a strong intensity-dependent effect in the log-ratios, the brain-clone signals became highly up-regulated at the same time as the strongly expressed liver genes were less affected by the intensity effect giving only little bias. The results also "indicate" that doing background subtraction is worse because it gives a higher MOR compared to when it is not applied. In reality this may or may not be true, but the indications are probably false due to reasons discussed above. This is an example where it is important to have an intensity-invariant precision measure. However, we still believe it is fair to compare MOR values for background and non-background corrected signals separately. We have not revisited this analysis with affine normalization procedures.

6.3 Paper C

Paper C presents the R package *R.oo*, which provides the object-oriented core used in the microarray aroma package, which all of our applied microarray research is based on. R.oo extends the S3 method dispatching mechanism in R with an object-oriented layer that provides references. By implementing support for reference variables, it allows a more user-friendly and memory efficient, but also more developer-friendly design. Reference variables are controversial, because they break the boundaries of the otherwise functional nature that the S language was intended for[5]. In spite of this, it provides us with the necessary language foundation for developing and maintaining the aroma package. Many other packages, not published here, rely on R.oo. It has proven to be very useful.

6.4 Paper D

Paper D conducts a detailed study of affine transformations of two-channel microarray data and their effects on the log-ratio transform. Many common normalization methods are investigated with respect to this transform. We conclude the study by suggesting a robust nonparametric normalization method for affinely transformed microarray data. With channels $\mathcal{C} = \{R, G\}$, $\alpha = a_R - \beta a_G$ and $\beta = b_R/b_G$ as in (11), we have for non-differentially expressed genes (without noise) that

$$y_{R,i} = \alpha + \beta y_{G,i}, \tag{21}$$

which is a special case of (16). Define the observation vector $\mathbf{y}_i = (y_{G,i}, y_{R,i})$. With noise (18) included, the points $\{\mathbf{y}_i\}_{i=1}^{I}$ are scattered around a line $L(\alpha, \beta)$ in $\mathbb{R}^2$ with intercept α and slope β. In order to estimate α and β, we define the objective function

$$Q(\alpha, \beta; \mathbf{y}) = \sum_{i=1}^{I} w_i r_i(\alpha, \beta; \mathbf{y}_i)^2 \tag{22}$$

where $r_i(\alpha, \beta; \mathbf{y}_i) > 0$ is the orthogonal Euclidean distance between $\mathbf{y}_i$ and the line $L(\alpha, \beta)$. The estimate of α and β is then

$$(\hat{\alpha}, \hat{\beta}) = \arg\min_{(\alpha,\beta)} Q(\alpha, \beta; \mathbf{y}). \tag{23}$$

[5] As a comment, we would like to note that the same functionalities that the R.oo core is based on is added to the data type environment in R v1.9.0, which in some sense should make R.oo less controversial.

Different weight functions minimize the orthogonal distances in different norms. For $w_i = 1$ minimization is done in L_2, just like principal component analysis (PCA), using $w_i = 1/(r_i(\hat{\alpha}, \hat{\beta}; \mathbf{y}_i) + \delta)$ minimization is done in L_1 (if we let $\delta \rightarrow 0^+$), and so on. The above minimization problem was implemented using iteratively reweighted PCA (IWPCA) and the method is very fast. Since the expression levels $x_{c,i}$ and $x_{d,i}$ are unknown, a_R and a_G are not individually identifiable. As described in Section 4.6.2, we obtain a *normalization* method by plugging the estimated parameters $\hat{\alpha}$ and $\hat{\beta}$ into (17) and then computing the estimated log-ratio $\widetilde{M}_i = \log_2(\tilde{y}_{c,i}/\tilde{y}_{d,i})$. In order to make the bias and scale parameters a_c and b_c identifiable, we suggest additional constraints in the paper. The method generalizes directly to multiple channels and even multiple arrays, which therefore makes so called *between-array (scale) normalization* unnecessary.

6.5 Paper E

Paper E gives evidence for existence of affine submeasurement functions in real microarray data. We investigate two different brands of scanners together with two different image-analysis applications and we find that the scanners introduce a significant bias in both channels. The results show that the bias is most likely introduced by the PMT and/or the following electronics such as the analog-to-digital converter. The image-analysis methods seem to introduce a small bias too, but of subsidiary order. As described in Section 4.6 above, by scanning the arrays at different PMT settings, using model (13) the channel biases can be uniquely identified. The estimation of bias and scale parameters is done by a modified version of the IWPCA method presented in Paper D. Model (13) is strongly supported by data. We refer to this method as the *multiscan calibration method*. Because the same array is scanned multiple times and the calibrated signals are averaged, the effective measurement noise is decreased and the effective dynamical range of the scanner is increased.

6.6 Paper F

Paper F describes the aroma (An R Object-oriented Microarray-Analysis environment) package discussed in Section 5. The package alone is written in more than 15000 lines of R code and has a 280-page manual with many examples. It is under continuous development.

6.7 Paper G

Paper G suggests a calibration method for spotted microarray data containing dilution series (replicated probes of various concentrations). The existence of dilution series makes it possible to identify a calibration function, instead of just a normalization function. The method is based on an affine heteroscedastic model. Let $y_{c,i,j}$ be the observed expression level for replicate j of gene i in channel c. We assume that[6]

$$y_{c,i,j} = a_c + b_c b_{i,j} x_{c,i} + \varepsilon_{c,i,j};\ \forall c, i, j \tag{24}$$

where a_c is the channel bias, b_c is the channel gain with $b_1 = 1$, $b_{i,j}$ is the probe concentration for replicate j of gene i, and $\varepsilon_{c,i,j}$ is independent noise with $E[\varepsilon_{c,i,j}] = 0$ and $V[\varepsilon_{c,i,j}] = \sigma^2_{c,i,j}$. A variance function that is quite flexible and at the same time generates feasible parameter estimates is

$$\sigma_{c,i,j} = k_c \sigma_i; \forall c, i \tag{25}$$

where k_c is the relative standard deviation in channel c with $k_1 = 1$ for identifiability. Recalling Section 4.7, contrary to a model with noise structure as in (19) and [11], this model assumes that the standard deviation is the same for all replicates regardless of spot concentration at the same time as it allows it to vary between genes. The model parameters are estimated with maximum likelihood (ML) techniques utilizing profile-likehood based grid search and PCA. In the paper, we study properties of bias parameter estimates $\hat{a}_c$ for microarray data simulated from model (24)-(25). We compute confidence intervals and standard errors for estimates of a_c using both bootstrap and analytical methods. The latter are based on normal approximations of the parameter estimates and computation of Fisher information of the structural parameters $\{a_c\}$, treating all other parameters as nuisance parameters. The estimated a_c is used for calibration of the probe signals. As a direct implication of the model design, estimates of the relative expression levels of the dilution series (genes) follow. For two channels, this calibration can either be defined through the replicate-specific log-ratios

$$\hat{M}_{i,j} = \log_2 \frac{y_{2,i,j} - \hat{a}_2}{y_{1,i,j} - \hat{a}_1} \tag{26}$$

[6] Note the differences in notation here and in the paper.

or the gene-specific version

$$\hat{M}_i = \log_2 \frac{\sum_j w_{i,j}(y_{2,i,j} - \hat{a}_2)}{\sum_j w_{i,j}(y_{1,i,j} - \hat{a}_1)}. \tag{27}$$

Here $w_{i,j}$ are weights chosen by the user, for instance $w_{i,j} = 1$ or $w_{i,j} = w(b_{i,j})$, in case the latter are known. We also estimate relative expression levels $\beta_{c,i} = (b_c b_{i,j} x_{c,i})/(b_1 b_{i,j} x_{1,i}) = b_c(x_{c,i}/x_{1,i})$. For two channels, this defines another calibration method

$$\hat{M}_i = \log_2(\hat{\beta}_{2,i}). \tag{28}$$

The log-ratios (26)-(28) are estimates of the true log-ratios up to an additive constant $\log_2 b_2$. Compared to (27), (28) has the advantage that standard errors and confidence intervals for $\hat{M}_i$ can be provided. These may be computed using either parametric bootstrap or analytical methods based on Fisher information and normal approximation. We also compare different types of dilution series $b_{i,j}$. Finally, we investigate robustness toward model misspecification in that we simulate data from (24), but with a variance function different from (25) and close to (19).

References

[1] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular Biology of the Cell*. Garland Science, fourth edition, March 2002.

[2] Anders Andersson, Johanna Keskitalo, Andreas Sjödin, Rupali Bhalerao, Fredrik Sterky, Kirsten Wissel, Karolina Tandre, Henrik Aspeborg, Richard Moyle, Yasunori Ohmiya, Rishikesh Bhalerao, Amy Brunner, Petter Gustafsson, Jan Karlsson, Joakim Lundeberg, Ove Nilsson, Göran Sandberg, Steven Strauss, Björn Sundberg, Mathias Uhlen, Stefan Jansson, and Peter Nilsson. A transcriptional timetable of autumn senescence. *Genome Biol*, 5(4):R24, 2004.

[3] Henrik Bengtsson. R Coding Conventions (draft), 2002. Published online: http://www.maths.lth.se/help/R/RCC/.

[4] B.P. Durbin, J.S. Hardin, D.M. Hawkins, and D.M. Rocke. A variance-stabilizing transformation for gene-expression microarray data. *Bioinformatics*, 18:S105–S110, 2002.

[5] Wolfgang Huber, Anja von Heydebreck, Holger Sültmann, Annemarie Poustka, and Martin Vingron. Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 1:1–9, 2002.

[6] Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

[7] Kim Johnson and Simon Lin. QA/QC as a pressing need for microarray analysis: meeting report from CAMDA'02. *Biotechniques*, Suppl:62–3, Mar 2003.

[8] D.J. Lockhart, H. Dong, M.C. Byrne, M.T. Follettie, M.V. Gallo, M.S. Chee, M. Mittmann, C. Wang, M. Kobayashi, H. Horton, and E.L. Brown.

Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nature Biotechnology*, 14(13):1675–80, Dec 1996.

[9] Harvey F. Lodish, Arnold Berk, Paul Matsudaira, Chris A. Kaiser, Monty Krieger, Matthew P. Scott, S. Lawrence Zipursky, James Darnell, and Harvey Lodish. *Molecular Cell Biology*. W.H. Freeman & Company, third edition, August 2003.

[10] Ingrid Lönnstedt and Terence P. Speed. Replicated microarray data. *Statistical Sinica*, 12(1), 2002.

[11] David M. Rocke and Blythe Durbin. A model for measurement error for gene expression arrays. *Journal of Computational Biology*, 8(6):557–569, 2001.

[12] David M. Rocke and Stefan Lorenzato. A two-component model for measurement error in analytical chemistry. *Technometrics*, 37(2):176–184, May 1995.

[13] Lao H. Saal, Carl Troein, Johan Vallon-Christersson, Sofia Gruvberger, Åke Borg, and Carsten Peterson. BioArray Software Environment (BASE): a platform for comprehensive management and analysis of microarray data. *Genome Biol*, 3(8):SOFTWARE0003, Jul 2002.

[14] Mark Schena. *Microarrays Analysis*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2003.

[15] Mark Schena, Dari Shalon, Ronald W. Davis, and Patrick O. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270(5235):467–470, October 1995.

[16] Gordon K. Smyth, Yee Hwa Yang, and Terry Speed. Statistical issues in cDNA microarray data analysis. *Methods Mol Biol*, 224:111–36, 2003.

[17] Terry Speed, editor. *Statistical Analysis of Gene Expression Microarray Data*. Interdisciplinary Statistics. Chapman & Hall/CRC, 2003.

[18] Bioconductor Core Team. Bioconductor - open source software for bioinformatics. http://www.bioconductor.org/, 2002.

[19] Jeroen van de Peppel, Patrick Kemmeren, Harm van Bakel, Marijana Radonjic, Dik van Leenen, and Frank C P Holstege. Monitoring global messenger RNA changes in externally controlled microarray experiments. *EMBO Rep*, 4(4):387–93, Apr 2003.

[20] Yee Hwa Yang, Sandrine Dudoit, Percy Luu, David M. Lin, Vivian Peng, John Ngai, and Terence P. Speed. Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucelic Acids Research*, 30(4):e15, Feb 2002.

A

Paper A

Identifying differentially expressed genes in cDNA microarray experiments: making aging visible

H. Bengtsson[1], B. Calder[2], I.S. Mian[3], M. Callow[4], E. Rubin[4], T.P. Speed[56]

[1] Mathematical Statistics at the Centre for Mathematical Sciences, Lund University, Lund
[2] Health Science Center at San Antonio, The University of Texas, Texas
[3] Genome Sciences Department, Lawrence Berkeley National Laboratory, Berkeley
[4] Radiation Biology and Environmental Toxicology, Life Sciences Division, Lawrence Berkeley National Laboratory, Berkeley
[5] Department of Statistics, University of California, Berkeley
[6] Division of Genetics and Bioinformatics, Walter and Eliza Hall Institute of Medical Research, Melbourne

This is a version of an online report accompanying the Science's SAGE KE discussion forum with the same title.

1 Introduction

Transcription profiling monitors simultaneously the abundance of tens of thousands of transcripts in a biological sample. In biogerontology, such profiling is but one arrow in the quiver required to ascertain how the physiological vigor of an organism declines over time. Currently, expression profiling technology comes in a variety of flavors including cDNA microarrays, high-density nylon membrane arrays, serial analysis of gene expression, short or long oligonucleotide arrays, fiber optic arrays, and so on [17]. The exponential rise in the number of profiling-related MEDLINE articles is a testament to its popularity. If this experimental tool is to deliver on its promises, however, a growing list of statistical, computational, technological and biological issues will need to be addressed (for incomplete lists, see [18, 15]).

This discussion forum focuses on the elemental, but critical issue of statistical approaches to identifying differentially expressed genes given cDNA microarray data (it should be noted that many of the same issues arise with the other technologies). The nature and type of data to be analyzed as well as a need to distinguish between biologically and statistically significant differential expression highlights the multiplicity of potential answers to such a deceptively simple question.

The starting premise for the forum is that a suitable scientific question has been formulated, the appropriate study designed, standard operating procedures employed to extract mRNA samples from relevant specimens, the transcription profiling experiments performed, and images of hybridized microarrays acquired. Accounting for technological and/or experimental confounding factors can reduce observation noise, errors associated with the measurement process or technology itself, but has little influence on model noise, variation due to the stochastic nature of gene expression and the underlying biology. It is assumed that strategies for minimizing observation noise and data analysis are independent of the scientific question. That is, aging is sufficiently similar to cancer, development and other biological processes so that no "new" statistical methods need to be invented to analyze transcription profiling data from biogerontology studies. Thus, employing a study of mouse lipid metabolism in mice ([7]) as the framework within which to structure the discussion forum and to illustrate pertinent issues seems appropriate.

The background corrected fluorescent intensities as well as the eight original scanned images processed using the Spot software [1] for the ApoAI experiments can be downloaded from [8]. The statistical analyses described on this page were performed using the object-oriented add-on com.braju.sma [4] to the sma (Statistics for Microarray Analysis) package [6] written using the R language [13]. How it can be done in R is described in the R program [5].

The outline of this article is as follows. In Section 2, we describe the setup of the experiments of the study and in Section 3 we shortly discuss the problem of extracting signal from scanned images. In Section 4, we will identify systematic variations in data that do not have a biological source, but are believed to stem from different error sources along the cDNA microarray process. We will show how different normalization methods can correct for these artifacts. In Section 5, we discuss the problems and methods on how to identify differentially expressed genes. Throughout the article we use the ApoAI data set to exemplify the different ideas. In Section 6, we give a list of 40 genes that we identified to be differentially expressed. At the end, we also do a comparison between the genes that are found when Spot is used and those that are found when GenePix is used.

2 Experimental setup

The goal of the study of mouse lipid metabolism in mice was to identify genes with altered expression in two mouse models with very low HDL cholesterol levels (treatment groups) compared to inbred control mice. The first mouse model compared eight mice with the apolipoprotein AI (ApoAI) knocked-out with a control group consisting of eight "normal" C57B1/6 mice. The second mouse model compares in a similar way eight scavenger receptor BI transgenic mice (SRBI) with the same control group. To identify the altered gene expressions a cDNA microarray technique was used. For the first model, target cDNA was obtained from mRNA for all sixteen mice by reverse transcription and labeled using the red-fluorescent dye Cy5. The reference sample was prepared by pooling cDNA from the eight control mice and was labeled with the green-fluorescent dye Cy3. An analog setup was used for the second mouse model. The mix of target and reference cDNA was hybridized to microarrays containing $N = 5548$ DNA probes. The slides contained 6384 spots, but some of them were empty spots. Here we will focus on the comparison between the eight ApoAI mice ($K = 8$) and the pool of control mice without making use of the fact that a comparison between each of the eight control mice with a pool of the same mice was also done. To include the eight slides from the control group in the analysis both *two-sample* t-tests and *adjusted p-values* are required, an approach which is beyond the scope of this article. For an analysis of the ApoAI data set using two-sample t-tests, see [11].

3 Image analysis

Raw data for one cDNA microarray slide typically consists of two scanned 16-bit images. While scanning, a green and a red laser excite the Cy3 and Cy5 dyes, and the emitted light is registered in every location. From the two images acquired, at least four different measures for each spot are extracted. The foreground signals, R_{fg} and G_{fg}, are often measured as the mean or the median of the intensities of the pixels that are identified to belong to the spot. There are several ways to define what the background of a spot is and different definitions often result in different background estimates, R_{bg} and G_{bg}. All background identification methods strive to get a representative estimate of the background noise in the area of the spot, noise that is also expected to be added to the foreground signals. With a good estimate of the foreground and the background signals it is reasonable to believe that the background subtracted signals, $R = R_{\mathrm{fg}} - R_{\mathrm{bg}}$ and $G = G_{\mathrm{fg}} - G_{\mathrm{bg}}$, reflect the gene-expression levels. It is important to remember that the spots are not always easily identifiable regions in the images, but can often be weak, or smeared out as comets (as an effect from the wash-off of the poly-L-lysine). It also happens that two or more spots are so large that they overlap or that the microarray slides are contaminated with dust and scratches. For reasons like these it is important to have good methods for estimating the foreground and the background signals. Yang et al. [20] compare the different foreground background identification and estimation methods that are used by the most commonly used image analysis software, such as GenePix [2], QuantArray [14], ScanAlyze [12], and Spot [1]. The authors show that the latter gives better foreground and background estimates than the others.

4 Normalization

Before any method for identifying differentially expressed genes can be applied, data has to be normalized. Normalization is a process of removing systematic variation in microarray experiments, which affects the measured gene-expression ratios. The sources of these artifacts can be many and they often reveal themselves in different ways. For instance, a too large imbalance in the PMT (photomultiplier tube) voltage setting for the red and the green laser in the scanner will give a bias in the fluorescence intensities toward one of channels. Another reason for having such a bias could be due to different physical properties of the two dyes (green Cy3 and red Cy5 dye), such as differences in half-life, size and weight, and heat and light sensitivity. Other systematic variations could be due to variations in the quality of the cDNA clones printed on the cDNA microarray slides. Varying printing conditions, such as temperature and humidity, during the printing process might further contribute to the variability in the measured gene-expression levels, or, as shown below, there may be problems with the print-tips.

The normalization process can be divided into two main steps. The first step identifies and normalizes for artifacts within each slide. The *within-slide normal-*

ization is applied to each slide independently of the others. To be able to compare the measured gene-expression levels between replicated slides the *spread* of expression levels must be the same for all slides. This is taken care of by the second normalization step called *between-slide normalization*.

4.1 Transformation of variables

Since the quantity of interest is the relative gene-expression level and R and G typically range from 0 to 65535, we transform data by taking the logarithm of the ratio between the red and the green signals, $M = \log_2(R/G)$. Another interesting quantity is the average log-intensity of the two signals, $A = 1/2 \cdot \log_2(RG)$. The rational for the factor $1/2$ is that A will have the same natural range as $\log_2 R$ and $\log_2 G$ and log-base two is used because data is originally binary. It is easy to show that this transform is reversible (and therefore the information contained in R and G is preserved). We have that

$$\begin{gathered} M = \log_2(R/G), \quad A = 1/2 \cdot \log_2(RG) \\ \Longleftrightarrow \\ R = (2^{2A+M})^{1/2}, \quad G = (2^{2A-M})^{1/2}. \end{gathered} \tag{1}$$

Many cDNA microarray experiments are set up in such a way that most of the genes are expected to be non-differentially expressed. In these cases an M versus A scatter plot will be more revealing than an R versus G scatter plot, because most of the data points in an R versus G plot will be distributed around the diagonal line $R = G$ whereas the same data points in an M versus A plot will be distributed around the horizontal line $M = 0$. In Figure 1 an R versus G and an M versus A scatter plot for the same set of data are shown. The normalization methods discussed below are operating in the (A, M) space. Also, if the error in the two channels is multiplicative, that is, $R = c \cdot R_{\text{true}}$ and $G = c \cdot G_{\text{true}}$, the transform to M will remove this noise.

It is important to have in mind that all the normalization methods used below are based on the assumption that there is only a small fraction of outliers $(1 - 2\%)$. We believe that this assumption is true for the ApoAI data set.

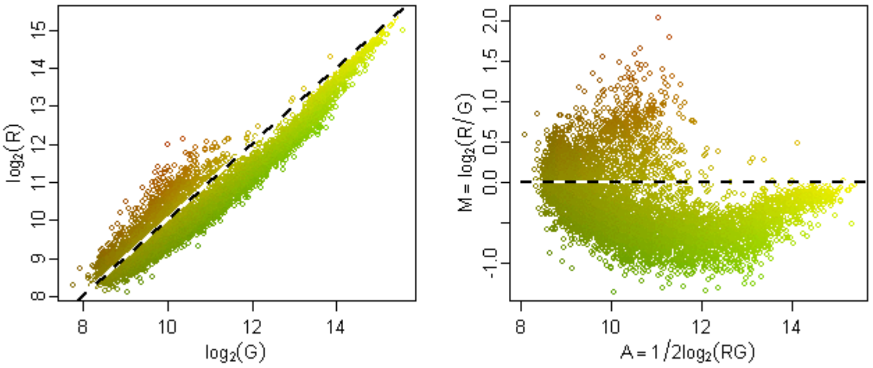


Figure 1: *Left*: An R versus G plot (log base 2) of the data points for the last slide in the ApoAI knock-out experiment. *Right*: The same data points plotted in a M versus A plot (log base 2). M is the log-ratio between the red and the green channel and A is the intensity. The dashed lines $R = G$ and $M = 0$ are where the signals in the two channels are equal.

4.2 Within-slide normalization

4.2.1 Global lowess normalization

Taking the mean (or the median) of the log-ratios for each slide we should, since we assume that most genes are non-differentially expressed, get values close to zero. A box plot of the M values for each of the eight ApoAI slides shows that this is not the case (cf. left plot in Figure 2). For slides 2, 3, and 5-8 the relative expression levels are biased toward the green channel (down-regulated genes) and for slides 1 and 4 there is a small bias toward the red channel (up-regulated genes). A first approach to normalize data would simply be to subtract the bias. However, the M versus A plot (for the third ApoAI slide), shows that the shift is not constant, but intensity-dependent (cf. right plot in Figure 2). The solid line in the plot is the lowess-fitted curve. The *lowess* function uses robust (not much affected by outliers) local linear regression to smooth scatter plots [9, 10]. All slides show this intensity-dependent pattern. This suggests that an *intensity-dependent*

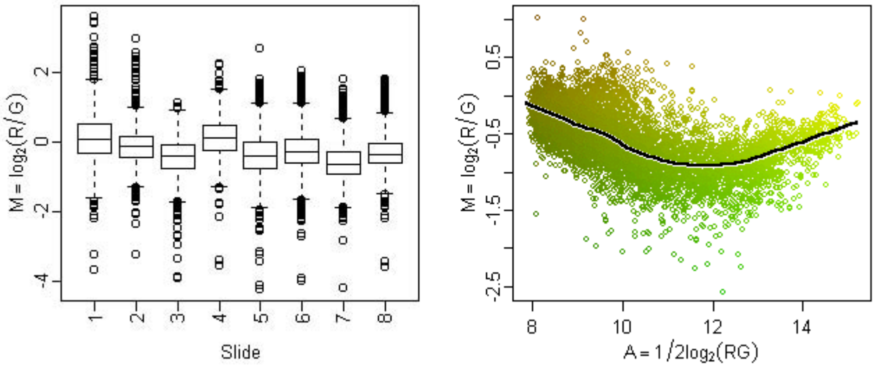


Figure 2: *Left*: Box-and-whisker plot showing the sample distribution of the log-ratios for each of the eight ApoAI slides. Since we assume most of the genes to be non-differentially expressed, ideally this plot should show a median around zero for all slides. *Right*: M versus A scatter plot for the third ApoAI slide with a lowess-fitted curve revealing that the log-ratios are intensity dependent.

normalization [8] is needed;

$$M_{\text{norm}} = M - c(A) \tag{2}$$

where $c(A)$ is the lowess fit to the M versus A plot.

4.2.2 Print-tip normalization

The printing step in the cDNA microarray process can also introduce systematic differences. In the ApoAI data set it is believed that there is an artifact present due to variation in the quality of the print tips [11, 21]. This could be due to different lengths of the print-tips making some of them leaving more cDNA on the slides than others, which in turn, due to dilution, could result in different concentration of cDNA in the wells on clone plates. Another reason could be deformation of some of the print-tip slits after hours of printing. For the ApoAI microarrays a print head of four-by-four print-tips were used ($J = 16$). Each print-tip printed a group of 399 spots, called a *print-tip group* (or grid). The print-tip groups are laid out on the slide from left-to-right and from top-to-bottom counting from the upper left corner to the lower right corner (cf. legend in Figure 4). Within each print-tip the spots are printed from left-to-right and from top-to-bottom. In other words, the sixteen spots in the upper left corner in the sixteen print-tip groups were printed at the same time, then the spots right next to them were printed and so forth. Even if the slides were printed with a well-organized clone library, this mapping of spots assures that any interesting genes are located "all over" the slide. For this reason we can assume that the "true" log-ratios have the same zero-mean distribution across the print-tip groups. A box plot of the log-ratios for the sixteen groups (cf. Figure 3) shows that there are systematic differences between the groups and especially that print-tip groups 13-16 are different from the others. A plot of the spatial distribution of the top 5% absolute log-ratios (cf. Figure 3) confirms this. This suggests that some kind of normalization within each print-tip group is needed. The (global) lowess normalization method above can be applied to each print-tip group individually. The sixteen lowess curves in Figure 4 confirm that the groups 13-16 are different from the others and that the

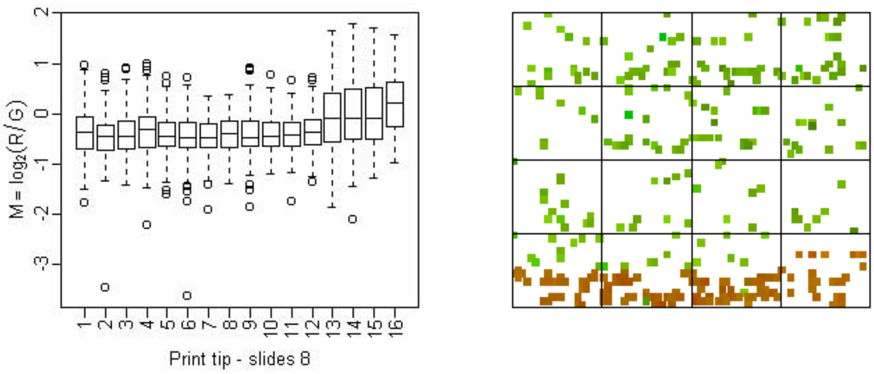


Figure 3: *Before normalization. Left*: Box-and-whisker plot of the data points in each of the sixteen print-tip groups. The lower four groups, 13-16, shows a different behavior than the rest. A perfect slide would have the same average and variance across the print-tip groups. *Right*: The spatial position of the top 5% absolute log-ratios ($|M|$) on the slide. Each small rectangle represents the log-ratio of a spot and the large four-by-four rectangles represent the print-tip groups. It is clear that there is, especially in the lower four groups, some kind of artifact.

differences are intensity dependent. The *print-tip normalization* [21] is

$$M_{j,\mathrm{norm}} = M_j - c_j(A_j);\; j = 1, 2, \ldots, J, \tag{3}$$

where J is the number of print-tip groups and $c_j(A_j)$ is the lowess fit of the log-ratios M_j and the log-intensities A_j of the spots belonging to print-tip group j. When doing a within-print-tip normalization a slide-intensity-dependent normalization as discussed in the previous section is unnecessary.

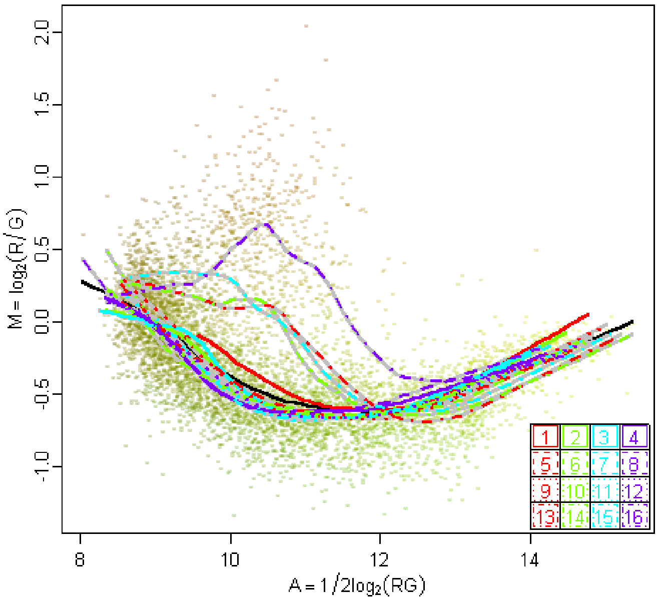


Figure 4: The M versus A plot with sixteen lowess-estimated curves, one for each of print-tip groups. The legend in the lower right corner shows how the print-tip groups are numbered, starting with one in the upper left group and counting left-to-right to sixteen in the lower right group. The signals from the four lowest print-tip groups, 13-16, have different properties than the others.

4.2.3 Scaled print-tip normalization

Doing a box plot and a spatial plot (not shown) similar to the ones in Figure 3 one gets that the print-tip normalization method not only removes the bias and the intensity dependency, but it also removes some of the spatial effects and makes the differences in variance between the groups smaller. However, there may still be differences in variance, which can be normalized by scaling the log-ratios within each print-tip group. The scale method used in [21] assumes that the log-ratios in each group, j, follow a distribution with zero mean and a variance σ_j^2. Not showing the details, it is straightforward to scale the print-tip groups so they get the same variance σ^2. To assure that the scaling is not affected by outliers the robust statistic MAD (median absolute deviation) is used to estimate the individual variances. The print-tip normalization method followed by a scaling step is called the *scaled print-tip normalization* method. The result of *scaled print-tip normalization* is shown in Figure 5.

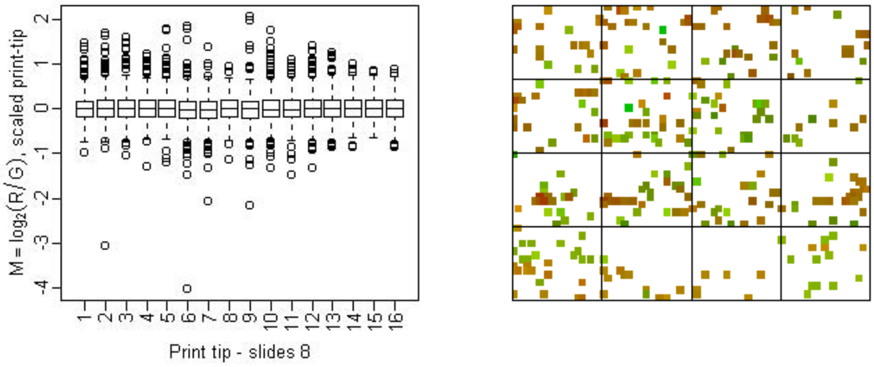


Figure 5: *After within-slide normalization*. *Left*: Box-and-whisker plot of normalized data for each of the sixteen print-tip groups. The scaled print-tip normalization method first adjusts data in each print-tip group so that their lowess lines become zero for all intensities. Second, it scales data within each group so that their variances become equal. *Right*: The spatial distribution on the slide of the top 5% absolute log-ratios,($|M|$), from ApoAI slide 8.

4.3 Between-slide normalization

Scaled print-tip normalization applied to all slides separately removes the intensity dependency of the log-ratios within the print-tip groups and this also removes the intensity dependency and the bias within the slides. However, the spread of the log-ratios might still vary from slide to slide and the same scaling technique that was applied to the print-tip groups may be used to scale variances between slides. *Between-slide normalization* makes it possible to compare the gene-expression levels measured by the different slides. The box plot in Figure 6 shows that scaled print-tip normalization followed by between-slide normalization makes the slides comparable (cf. Figure 2). The "average" slide is shown as an M versus A scatter plot in Figure 6. The bars over the axis labels denote the mean of M and the mean of A, respectively.

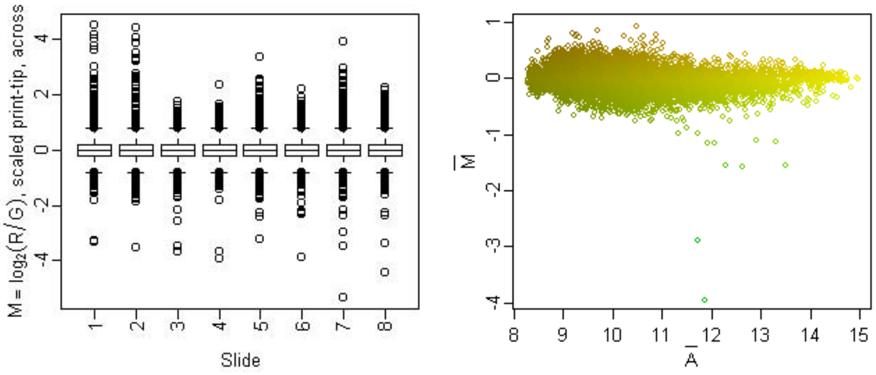


Figure 6: *Left*: Box-and-whisker plot showing the sample distribution of the log-ratios of the eight ApoAI slides after scaled print-tip normalization and between-slide normalization. *Right*: The "average" slide where the mean of the log-ratios and the mean of the intensities have been calculated for all spots across the slides.

5 Identifying differentially expressed genes

5.1 Absolute log-ratios and the t-statistic

The most simple way to select genes being differentially expressed is to stratify on the absolute average log-ratios, e.g. by choosing genes that have $|M| > M_{\text{min}}$.

The main argument against using the average M value to decide if a gene is differentially expressed or not, is that a large mean might be driven by an outlier, something which is not rare for cDNA microarray measurements. Thus, a better statistic to be used is the traditional *t-statistic*:

$$t_n = \bar{M}_n/\mathrm{SE}_n;\ n = 1, 2, \ldots, N \tag{4}$$

where $\bar{M}_n$ is the (sample) mean of the log-ratios for gene n; $(M_{n,1}, \ldots, M_{n,K})$. $\mathrm{SE}_n = s_n/K^{1/2}$ is the standard error of the mean, where s_n is the (sample) standard deviation of the log-ratios for gene n. Since we do not care if the genes are up or down regulated, we look at the absolute t value, $|t|$. For two genes with the same mean of log-ratios, the gene with the smaller standard error will get the larger $|t|$ value. The rational for this is that if repeated measurements of a gene-expression ratio give very similar results, i.e. the standard error is small, we should trust these measurements more than in the case where they differ a lot.

There are problems with the t-statistics. A gene with a small absolute mean might get a large $|t|$ value only because of a small SE, which might not be the true reflection of the gene. Remember that we are not only comparing t values from two genes but from several thousands of genes. For this reason, *by chance* there will be genes that have larger $|t|$ values because they have small SE, not because they are differentially expressed. An ad hoc way to solve this problem is to discount genes with a small absolute mean whose standard errors are small. A small SE could be defined as SEs that are among the bottom 1%. Tusher et al. [19] presented another solution to the problem where they add a constant to each SE. Their statistic is referred to as the *S-statistic*.

5.2 The B-statistic

A better and more correct method for dealing with the above problem and for identifying differentially expressed genes is an empirical Bayes method developed by Lönnstedt et al. [16]. The authors introduce a new statistic called the *B-statistic*. The B value calculated for gene n is a log-posterior odds ratio for the gene to be differentially expressed, against to be non-differentially expressed, given the observed log-ratios for *all* genes and *all* experiments:

$$B_n = \log \frac{P(\text{gene } n \text{ is differentially expressed}|\{M_{n,k}\})}{P(\text{gene } n \text{ is non-differentially expressed}|\{M_{n,k}\})} \tag{5}$$

where $n = 1, 2, \ldots, N$. The probabilistic model behind this B-statistic assumes that the K replicated log-ratio measurements for gene n, $M_{n,k}; k = 1, 2, \ldots, K$, are normally distributed with mean μ_n and variance σ_n^2. However, the mean and the variance are not treated as fixed parameters, but as random variables themselves, and therefore we look at the distribution of $M_{n,k}$ as a conditional distribution $M_{n,k}|\mu_n, \sigma_n^2 \sim N(\mu_n, \sigma_n^2)$. The assumption is still that most genes are non-differentially expressed, that is, they have $\mu_n = 0$, but a small proportion p of genes have a $\mu_n \neq 0$. The expression above can now be written $B_n = \log[P(\mu_n \neq 0|\{M_{n,k}\})/P(\mu_n = 0|\{M_{n,k}\})]$. Assuming special conjugate-priori distributions for the mean and the variances (for details see [16]), the authors get the full expression for B_n to be:

$$B_n = \log \frac{p}{1-p} \cdot \frac{1}{(1+Nc)^{-1/2}} \cdot \left[\frac{a + s_n^2 + \bar{M}_n^2}{a + s_n^2 + \bar{M}_n^2/(1+Nc)} \right]^{\nu + N/2} \tag{6}$$

where $\bar{M}_n$ is the sample mean and s_n^2 is sample variance of the K values for gene n with N as the number of genes. Leaving out the details, a, ν and c are *hyperparameters* that are estimated from the log-ratios from *all* genes and *all* slides. Finally, p is the proportion of differentially expressed genes and p is normally the only parameter to be tuned. The B-statistic works in a similar fashion as the S-statistic, but the former gives a slightly smaller number of false positives and false negatives [16].

5.3 Number of replicates needed

Before presenting the results, a relevant and important question to be asked is how many replicates are that needed to get trustful results. We did all data analysis discussed above for the cases where $n = 2, 4, 6, 8$ slides were used. Since the number of slides does not affect within-slide normalization we did not have to repeat that step. However, we had to do the between-slide normalization for each n. The top 40 genes when two, four, six and eight slides were used are listed in Table 1. The genes are ranked according to their B values. For $n = 2$ there were no genes with a positive log-posterior odds ratio among the top 40 genes. For $n = 4$ there were 12 with positive B values, and for $n = 6$ and $n = 8$ all top 40 genes had positive B values. More interestingly, when data from two slides are used, we can only identify 10 out of the top 40 genes found when all slides were used. When four slides were used, 17 out of 40 were found, and when

	Number of slides incl. in the analysis					Number of slides incl. in the analysis			
Rank	$n = 2$	$n = 4$	$n = 6$	$n = 8$	Rank	$n = 2$	$n = 4$	$n = 6$	$n = 8$
1	*__2149__	*__2149__	*__2149__	*__1496__	21	3530	6124	__4951__	*___540___
2	*__1496__	*__1496__	*__4139__	*__2149__	22	1428	5417	*__1739__	___1257___
3	*___2537___	*__4139__	*__1496__	*__4941__	23	1270	483	__6130__	___5821___
4	1337	6050	*__4941__	*__4139__	24	3942	5749	__3153__	___5703___
5	*__4139__	__6061__	6124	__6043__	25	4922	823	*5356	___6118___
6	4357	__6130__	__6061__	__761__	26	3339	__6045__	___2314___	___5353___
7	3023	*__1739__	5345	__4930__	27	4354	6327	5731	__2296__
8	6033	*__4941__	__761__	__6129__	28	4942	5418	6216	__4925__
9	__6061__	5345	__6043__	*__1739__	29	5731	5116	___2186___	__6042__
10	5287	5731	*___2537___	__5719__	30	5625	2922	___6116___	__4884__
11	5345	__6043__	1658	__3153__	31	6040	__4930__	4960	__6045__
12	1049	*___2537___	__6129__	__6061__	32	6050	__4951__	3151	__6130__
13	3251	4942	1270	__4875__	33	3261	3729	5418	__5746__
14	__6130__	__6129__	6050	___6116___	34	26	__5719__	___5751___	__6109__
15	1494	5729	__4930__	*___2537___	35	5660	5335	2963	__5944__
16	*__1739__	1658	__6045__	___4530___	36	27	__4875__	1337	__4882__
17	__761__	2963	473	___2186___	37	2675	4922	___4529___	__4874__
18	5730	5700	__5719__	___5751___	38	3552	2748	__5673__	__5358__
19	__4951__	__761__	1755	___4529___	39	__6043__	4148	4148	__5673__
20	1911	___2314___	__4875__	___2314___	40	483	6222	___5703___	__4951__

Table 1: *Genes found as a function of the number of replicates*. The table is listing the top 40 (0.6%) ApoAI differentiated genes found when the number of slides used in the analysis is $n = 2, 4, 6, 8$, respectively. In each column, the genes are ranked by their B-statistic. The number listed are the genes' spot numbers as they occur on the microarray slides. The genes marked with a star (*) are the eight genes listed in [11]. The color and style coding is only used to help locate the same gene across the columns. All genes found when $n = 8$ slides were used are **bold-faced**. To simplify the identification of these genes in the $n = 2$, $n = 4$ and $n = 6$ columns, they are also color coded and some of them are in italic. Genes not found in the $n = 8$ column are non-bold and colored in silver.

six slides were used, 24 out of 40 were found. It must be mentioned that the selection of two, four and six slides was done by taking the first two slides, then the first four and finally the first six slides as they occur in the data set. A more rigorous approach would be to use a *bootstrap technique*, which repeatedly samples two (four or six) slides from the set of eight slides and then takes the average of the B-values for each gene.

6 Results

6.1 Top forty differentially expressed genes in the ApoAI data set

Using the B-statistic on the ApoAI data set we identify a set of genes that we believe to be differentially expressed. The genes are listed Table 2, but can also be downloaded in a computer-readable format from [3]. Almost all the identified genes are *down regulated*, which makes sense since we are looking at a knock-out model. A more detailed discussion of the biological relevance of these genes are left to the SAGE KE discussion forum. The B values were calculated assuming 1% of the genes to be differentially expressed ($p = 0.01$). If we instead assume 0.5% of the genes to be differentially expressed ($p = 0.005$) we get that the 40 genes with largest log-posterior odds ratio are still ranked the same as in the $p = 0.01$ case. A B versus average M plot (cf. left plot in Figure 7) of the ApoAI data set shows that selecting genes based solely on the average log-ratios would not give the same set of genes as if the log-posterior odds ratio would be used. In the plot, the 40 (0.6%) genes with highest ranked B values are highlighted red. To the right in Figure 7, the same genes are highlighted, but now in an average M versus average A plot.

6.2 Comparison between GenePix and Spot

In addition to using the Spot software for extracting the signals in the cDNA microarray images, we used the GenePix software. We analyzed data from GenePix in exactly the same way as we did with data from Spot and we compared the genes identified. The genes with the top 40 largest B values found by GenePix and the genes with the top 40 largest B values found by Spot are listed in Table 3. Both Spot and GenePix ranked the gene of spot 1496 to be the most differentially expressed gene. Spot ranked 2149 second whereas GenePix ranked spot 4941 second and so on. In GenePix's top 40 list there are 15 genes that are found in Spot's top 40 list, or equivalent in GenePix's top 40 list, there are 25 genes that are *not* found in Spot's top 40 list. Descriptions of these 25 genes are listed in the last column in Table 3.

Finally, looking at the B values, Spot seems to produce larger B values than GenePix. This can be interpreted as suggesting that the quality of data (signal to noise) from Spot is slightly higher than that from GenePix. However, a preferred comparison would be one based on their respective abilities to correctly identify

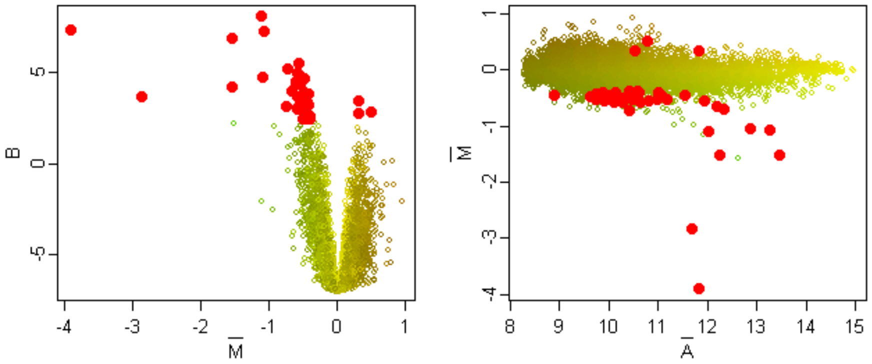


Figure 7: *Left*: The log-posterior odds ratio versus the average log-ratios. The B-statistic is the log-posterior odds ratio for each gene, which here is the same as each spot, to be differentially expressed. It is clear from this plot, that identification of differentially expressed genes based only on the average log-ratios would not give the same genes as if it was based on the log-posterior odds ratio. Small $|M|$ values tend to correspond to small B values, but the reverse is not true. The genes with the top 0.6% largest B-statistics are highlighted red. We used the parameter value $p = 0.01$, i.e. we expect about 1% of the genes to be differentially expressed. *Right*: A scatter plot of the average log-ratios versus the average intensities with the same identified genes highlighted.

genuinely differentially expressed genes, and on their false positive rates. Such a comparison is not possible at this point.

Rank	spot	M	A	B	R	G	id	Description
1	1496	-1.11	12.0	**8.12**	2828	6121	484183	est
2	2149	-3.91	11.8	**7.31**	934	14049	1077520	Apo AI, lipid-Img
3	4941	-1.07	12.9	**7.25**	5226	10941	737183	similar to yeast sterol desaturase, lipid-Img
4	4139	-1.52	12.3	**6.87**	2909	8373	374370	EST, Weakly similar to C-5 STEROL DESATURASE [Saccharomyces cerevisiae], lipid-UG
5	6043	-0.54	9.8	**5.46**	712	1038	870879	5'. gi|2186618|gb|AA461727|AA461727 [2186618], AA461727
6	761	-0.72	12.4	**5.18**	4068	6710	519093	Glucosidase, alpha, acid
7	4930	-0.57	9.9	**4.93**	796	1182	493497	ESTs, Highly similar to AROMATIC-L-AMINO-ACID DECARBOXYLASE [Rattus norvegicus], Brain-UG
8	6129	-0.54	11.2	**4.80**	1954	2834	570543	ESTs, Highly similar to C-8 STEROL ISOMERASE [Magnaporthe grisea], lipid-UG
9	1739	-1.09	13.3	**4.74**	6758	14418	483614	Apo CIII, lipid-Img
10	5719	-0.54	10.5	**4.72**	1191	1732	480221	Mouse steroid 21-hydroxylase A (21-OHase A) gene, lipid-UG
11	3153	-0.47	11.6	**4.69**	2551	3539	540161	Mus musculus alanine:glyoxylate aminotransferase (Agxt1) mRNA, complete cds
12	6061	-0.59	10.3	**4.49**	1012	1526	871167	5'. gi|2187584|gb|AA462693|AA462693 [2187584], AA462693
13	4875	-0.60	10.1	**4.40**	897	1359	437224	Isl1, AA003609
14	6116	-0.53	10.2	**4.33**	953	1376	526650	Murine GABA-A receptor delta-subunit gene, Brain-Img
15	2537	-1.53	13.5	**4.16**	6655	19208	483614	ESTs, Highly similar to APOLIPOPROTEIN C-III PRECURSOR [Mus musculus], lipid-UG
16	4530	-0.66	12.2	**3.94**	3758	5928	444794	BRAIN PROTEIN D3, Brain-UG
17	2186	-0.47	8.9	**3.90**	404	558	MDB0345	MDB0345, MDB0345
18	5751	-0.50	9.8	**3.83**	728	1031	918176	NEURONAL ACETYLCHOLINE RECEPTOR PROTEIN, ALPHA-5 CHAIN, Brain-Img
19	4529	-0.41	9.9	**3.79**	818	1089	443390	EST, Moderately similar to APOLIPOPROTEIN B-100 PRECURSOR [Homo sapiens], lipid-UG
20	2314	-0.56	11.9	**3.68**	3255	4816	372643	Uncoupling protein 2, mitochondrial
21	540	-2.85	11.7	**3.67**	1231	8890	439353	EST, Highly similar to APOLIPOPROTEIN A-I PRECURSOR [Mus musculus], lipid-UG
22	1257	-0.48	9.6	**3.61**	672	938	870815	5'. gi|2186896|gb|AA462005|AA462005 [2186896], AA462005
23	5821	0.33	10.5	**3.45**	1667	1325	MDB1531	MDB1531, MDB1531
24	5703	-0.47	10.5	**3.38**	1197	1661	1196370	Grb2 (downstream of rec kinase homol), AA794463
25	6118	-0.42	10.2	**3.34**	982	1318	552012	FK506-BINDING PROTEIN PRECURSOR, heart-UG
26	5353	-0.41	10.6	**3.18**	1361	1807	1312454	MEK Kinase 3, heart-Img
27	2296	-0.73	10.4	**3.11**	1059	1763	475851	est
28	4925	-0.57	10.8	**3.07**	1476	2191	482240	Mus musculus APC-binding protein EB2 mRNA, partial cds, heart-UG
29	6042	-0.51	10.3	**2.98**	1062	1510	870875	5' similar to TR:G972043 G972043 MRNA; EXPRESSED SEQUENCE TAG ;. gi|2186616|gb|AA461725|AA461725 [2186616], AA461725
30	4884	-0.55	11.0	**2.96**	1663	2440	467322	Id like, AA036390
31	6045	-0.58	10.7	**2.94**	1322	1978	870887	5'. gi|2186621|gb|AA461730|AA461730 [2186621], AA461730
32	6130	-0.53	10.3	**2.93**	1069	1540	572995	Mouse Ki-ras cellular oncogene exon 1 from Y1 adrenal tumor cells (and joined CDS), heart-UG
33	5746	-0.47	10.0	**2.89**	850	1179	846842	Alpha-1B Adrenergic Receptor, heart-Img
34	6109	0.51	10.8	**2.83**	2132	1502	516686	Nkx2.6, AA108980
35	5944	0.33	11.8	**2.73**	4085	3250	478048	NEURONAL PROTEIN 3.1
36	4882	-0.39	10.4	**2.60**	1210	1583	466584	Ntrk3 (Neurotrophic RTK), AA030917
37	4874	-0.39	10.6	**2.42**	1361	1785	427319	ephrinB1, AA003297
38	5358	-0.42	11.0	**2.40**	1799	2413	1362186	NGFI-A BINDING PROTEIN 1, Brain-Img
39	5673	-0.45	9.8	**2.39**	742	1014	437516	EST- Contaxin precursor e-5, AA003596
40	4951	-0.49	11.1	**2.39**	1851	2592	876791	beta-3-adrenergic receptor, Brain-Img

Table 2: *Differentially expressed genes*. The columns are 1) the rank (from B-values), 2) the spot number, 3) the log-ratio, 4) the intensity, 5) the log-posterior odds ratio, 6) the Cy5 intensity, 7) the Cy3 intensity, 9) the gene id, and 9) a short description of the gene/est. The list of genes can be downloaded in a computer-readable format from [18].

	Spot top 40				GenePix top 40				
Rank	spot	M	A	B	spot	M	A	B	Gene ranked highly by GenePix but not Spot
1	*1496	-1.11	12.0	**8.12**	*1496	-1.06	11.5	**8.24**	
2	*2149	-3.91	11.8	**7.31**	*4941	-1.44	13.0	**6.95**	
3	*4941	-1.07	12.9	**7.25**	603	1.64	5.4	**5.55**	MDB0670, R75477
4	*4139	-1.52	12.3	**6.87**	6130	-0.71	9.8	**4.89**	
5	6043	-0.54	9.8	**5.46**	6124	-0.96	8.4	**4.75**	Androgen receptor, Brain-UG
6	761	-0.72	12.4	**5.18**	3489	0.73	9.0	**4.57**	est
7	4930	-0.57	9.9	**4.93**	4875	-0.85	9.7	**4.32**	
8	6129	-0.54	11.2	**4.80**	6217	0.67	10.5	**4.25**	5’ novel gene, H64597
9	*1739	-1.09	13.3	**4.74**	5032	0.93	10.0	**4.13**	MDB0488, R75338
10	5719	-0.54	10.5	**4.72**	4530	-0.64	12.1	**4.04**	
11	3153	-0.47	11.6	**4.69**	*1739	-1.17	13.5	**3.82**	
12	6061	-0.59	10.3	**4.49**	*540	-3.37	11.2	**3.78**	
13	4875	-0.60	10.1	**4.40**	761	-0.74	12.3	**3.55**	
14	6116	-0.53	10.2	**4.33**	6221	0.79	10.6	**3.48**	novel gene, AA505401
15	*2537	-1.53	13.5	**4.16**	5700	-0.93	9.4	**3.46**	Jagged1 (Alagille syndrome), AA619107
16	4530	-0.66	12.2	**3.94**	5274	-0.61	9.5	**3.39**	Jak3, AA023709
17	2186	-0.47	8.9	**3.90**	6284	-0.57	10.2	**3.39**	est
18	5751	-0.50	9.8	**3.83**	4855	-0.94	9.4	**3.35**	5’. gi|2187342|gb|AA462451|AA462451 [2187342], AA462451
19	4529	-0.41	9.9	**3.79**	4961	-0.70	8.7	**3.33**	Myosin Binding Protein C, heart-Img
20	2314	-0.56	11.9	**3.68**	1514	-0.80	7.9	**3.20**	est
21	*540	-2.85	11.7	**3.67**	563	-0.97	11.2	**2.94**	FATTY ACID-BINDING PROTEIN, EPIDERMAL, lipid-UG
22	1257	-0.48	9.6	**3.61**	1519	-0.72	8.6	**2.80**	est
23	5821	0.33	10.5	**3.45**	5283	-0.67	9.5	**2.76**	Hkr-t1, AA110661
24	5703	-0.47	10.5	**3.38**	2803	0.88	8.4	**2.69**	5’ similar to PIR:A56136 A56136 jagged protein precursor - rat ;. gi|1287598|gb|W13561|W13561 [1287598], W13561
25	6118	-0.42	10.2	**3.34**	2235	1.40	8.2	**2.68**	5’ similar to gb:X64229_cds1 DEK PROTEIN (HUMAN), AA445930
26	5353	-0.41	10.6	**3.18**	5349	-0.71	9.2	**2.67**	G PROTEIN-COUPLED RECEPTOR KINASE GRK5, Brain-Img
27	2296	-0.73	10.4	**3.11**	4830	-0.47	11.6	**2.61**	5’. gi|2192760|gb|AA466620|AA466620 [2192760], AA466620
28	4925	-0.57	10.8	**3.07**	4862	-0.98	9.1	**2.59**	5’ similar to WP:F32D8.4 CE05783 LACTATE DEHYDROGENASE ;. gi|2187569|gb|AA462678|AA462678 [2187569], AA462678
29	6042	-0.51	10.3	**2.98**	*2537	-1.42	13.7	**2.57**	
30	4884	-0.55	11.0	**2.96**	*5356	-1.98	13.0	**2.52**	CATECHOL O-METHYLTRANSFERASE, MEMBRANE-BOUND FORM, Brain-Img
31	6045	-0.58	10.7	**2.94**	6061	-0.85	9.8	**2.45**	
32	6130	-0.53	10.3	**2.93**	*2149	-4.79	11.3	**2.33**	
33	5746	-0.47	10.0	**2.89**	5244	-0.90	8.8	**2.32**	5’. gi|2186615|gb|AA461724|AA461724 [2186615], AA461724
34	6109	0.51	10.8	**2.83**	722	-0.64	10.3	**2.31**	Mus musculus thrombospondin-4 mRNA, complete cds
35	5944	0.33	11.8	**2.73**	5358	-0.80	10.3	**2.21**	
36	4882	-0.39	10.4	**2.60**	6045	-0.72	10.3	**2.21**	
37	4874	-0.39	10.6	**2.42**	5268	-0.68	9.0	**2.11**	Ash1 homolg4, AA014730
38	5358	-0.42	11.0	**2.40**	2296	-1.02	9.7	**2.02**	
39	5673	-0.45	9.8	**2.39**	3585	0.78	9.7	**2.01**	B actin, 1030797
40	4951	-0.49	11.1	**2.39**	*4139	-1.54	12.2	**1.98**	

Table 3: *Genes ranked highly by Spot versus genes ranked highly by GenePix.* In column 2 are the spot number for the genes found using Spot for the image analysis. In column 6 are the spot number for the genes found by using GenePix for the image analysis. Except for the image analysis the rest of the data analysis was identical. In the last column are the descriptions of the genes that were highly ranked by GenePix but not by Spot listed. For the rest of the description, both for the GenePix and the Spot genes, see Table 2. The color and style coding is done in the same fashion as in Table 1.

References

[1] CSIRO Australia. *Spot user's manual.* CSIRO Mathematical and Information Sciences, Image Analysis Group, July 2003. http://spot.cmis.csiro.au/spot/.

[2] Inc. Axon Instruments. Genepix software. http://www.axon.com/GN_Genomics.html, 2001.

[3] H. Bengtsson, B. Calder, I.S. Mian, M. Callow, E. Rubin, and T.P. Speed. List of 40 genes found to be differentially expressed in the ApoAI knockout experiments. http://www.maths.lth.se/matstat/bioinformatics/sageke/apoa1-genelist.txt, 2001. (a comma-separated ASCII file).

[4] Henrik Bengtsson. com.braju.sma - object-oriented microarray analysis in 100% R. http://www.maths.lth.se/help/R/, 2001.

[5] Henrik Bengtsson. Example R program that loads the data, normalizes it and identifies differentially expressed genes using the B-statistics method. http://www.maths.lth.se/matstat/bioinformatics/sageke/Rcode.txt, October 2001.

[6] Ben Bolstad, Sandrine Dudoit, Ingrid Lönnstedt, Natalie Roberts, and Jean Yee Hwa Yang. R package: Statistics for Microarray Analysis (sma). http://www.stat.berkeley.edu/users/terry/zarray/Software/smacode.html, 2001.

[7] M. Callow, S. Dudoit, E. Gong, T. Speed, and E. Rubin. Microarray expression profiling identifies genes with altered expression in HDL-deficient mice. *Genome Research*, 10(12):2022–9, December 2000.

[8] Matt Callow. The ApoAI data set. Genome Sciences Department, Lawrence Berkeley National Laboratory, Berkeley. http://www.stat.berkeley.edu/users/terry/zarray/Html/apodata.html, 1999.

[9] W.S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of American Statistics Association*, 74:829–836, 1979.

[10] W.S. Cleveland. LOWESS: A program for smoothing scatterplots by robust locally weighted regression. *The American Statistician*, 35:54, 1981.

[11] Sandrine Dudoit, Yee Hwa Yang, Matthew J. Callow, and Terence P. Speed. Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments. Technical Report 578, Department of Statistics, University of California at Berkeley, 2000.

[12] Michael Eisen. Scanalyze software. http://rana.lbl.gov/EisenSoftware.htm, 2001. Eisen Lab, Life Science Division, Lawrence Berkeley National Laboratory, Berkeley, California.

[13] Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

[14] PerkinElmer Inc. Quantarray analysis software. http://las.perkinelmer.com/, 2004. (former Packard Instrument Company).

[15] W. Li. Bibliography on microarray data analysis. http://www.nslij-genetics.org/microarray/. Rockefeller University.

[16] Ingrid Lönnstedt and Terence P. Speed. Replicated microarray data. *Statistical Sinica*, 12(1), 2002.

[17] L. Shi. DNA Microarray (Genome Chip) - Monitoring the Genome on a Chip. http://www.gene-chips.com/, 2001.

[18] Terry Speed. Statistical problems involving microarray data. http://www.stat.Berkeley.EDU/users/terry/zarray/Html/list.html, 2001. University of California, Berkeley.

[19] V.G. Tusher, R. Tibshirani, and G. Chu. Significance analysis of microarrays applied to the ionizing radiation response. *Proc Natl Acad Sci U S A*, 98(9):5116–21, Apr 2001.

[20] Yee Hwa Yang, Michael Buckley, Sandrine Dudoit, and Terry Speed. Comparison of methods for image analysis on cDNA microarray data. Technical Report 584, Department of Statistics, University of California at Berkeley, Nov 2000.

[21] Yee Hwa Yang, Sandrine Dudoit, Percy Luu, and Terence P. Speed. Normalization for cDNA microarray data. In Michael L. Bittner, Yidong Chen, Andreas N. Dorsel, and Edward R. Dougherty, editors, *Proceedings of SPiE*, volume 4266 of *Microarrays: Optical Technologies and Informatics*, pages 141–152, San Jose, California, June 2001. The International Society for Optical Engineering.

B

Paper B

Identification and normalization of plate effects in cDNA microarray data

Henrik Bengtsson

Mathematical Statistics at the Centre for Mathematical Sciences, Lund University, Box 118,
SE-221 00 Lund, Sweden. Email: hb@maths.lth.se.

ABSTRACT

By introducing a novel way of visualizing cDNA microarray data we have identified a new type of systematic variation, which we refer to as *plate effects*. We believe that plate effects are due to non-biological differences in the DNA clones spotted onto the microarray slides. By comparing the variability of replicated spots (both within and between slides) after performing 42 different normalization strategies, we claim that the plate effects are non-negligible and should be corrected for in addition to the already known intensity-dependent effects. For comparison between normalization strategies we introduce a robust genewise variability measure, which we call *measure of reproducibility*. In addition, for the data set studied, we find that not doing background correction improves the reproducibility significantly.

Keywords: cDNA microarrays; print-order plot; systematic variation; artifacts; plate effects; clone effects; normalization.

1 Introduction

The cDNA microarray technology is a relatively new technique for measuring (relative) gene-expression levels in two different cell lines simultaneously. The technique covers a huge spectrum of applications such as identification of oncogenes (genes up-regulated in cancer), time-series studies of cell lines under different environmental stress conditions, clustering of expression profiles to identify related genes etc. The cDNA microarray technique is also started to be used in clinic trials for diagnosis on a patient-to-patient basis. With the vast amount of biological data produced an increasing need for new statistical and computational methods is seen.

Robotically, using a so called *arrayer*, up to 40000 unique DNA (deoxyribonucleic acid) spots are printed onto a microscope glass slide. RNA from the two samples are purified separately, amplified and then turned into cDNA by reverse transcription. During reverse transcription the two solutions of cDNA are labeled with two different dyes, commonly the fluorescent dyes Cy3 and Cy5. Equal amounts of the two labeled cDNA solutions are co-hybridized to the DNA spots on the glass slide. The large number of cDNA strands and the competitive nature of hybridization make it possible to approximate the relative gene expression of a certain gene with the relative amount of Cy3 and Cy5 in the corresponding spot. The amounts of Cy3 and Cy5 in each spot are estimated by scanning the hybridized microarray slide at wavelengths 632 nm and 535 nm, which excite the Cy3 and Cy5 fluorescent dyes, respectively. The foreground and background signals of the spots are extracted and used in downstream analysis.

In the above process, there are several sources of error that introduce artifacts in data. In addition to problems of such steps as selecting sample tissues and extracting RNA, transcribing it into cDNA and so on, there are several other quality related problem. Dust sticking to the slides produces false signal peaks inside some spots, non-linearity between the two channels may favor one of the cell lines, spatial variation across a slide or systematic variation between replicated slides could additionally result in false conclusions. The list of artifacts is long, where some are well understood whereas some are still to be explained or identified.

In this paper we discuss what we believe is an artifact due to systematic variation between the DNA clone libraries and/or systematic variation between spots printed during the several hours long printing process. We refer to these artifacts as *plate effects*. As explained later, it is not obvious how to distinguish, a priori, between plate effects and effects due to differences in the two fluorescent labels. To identify the most dominant sources of systematic variations, we use a technique where different sequences of normalization methods, referred to as *normalization strategies*, are applied to data. Introducing a measure of reproducibility, we then search for the sequence of normalization steps that gives the least variable signals across replicated slides.

All methods discussed in this paper have been implemented using the R software [6] and the *com.braju.sma* package [1], which is an object-oriented extension to the *sma* package [3] and relies on the *R.oo* package part of the R.classes [2].

The data set analyzed is described in Section 2, followed by Section 3, which lists possible artifacts in data. Previously known systematic variations are discussed, but also a not commonly discussed effect, referred to as *plate effects*. Print-order plots are introduced. In Section 4 different methods to normalize for artifacts are discussed. Different methods for normalization of plate effects are given in detail. Section 5 defines 42 different normalization strategies to be compared as well as the *Measure of Reproducibility* (MOR) used to compare the different strategies. Results are given in Section 6. Section 7 concludes with a discussion of underlying reason for plate effects.

2 Data

Data for the eight cDNA microarray slides analyzed in this paper comes from the Matt Callow Lab at Lawrence Berkeley National Laboratories [4]. Their experimental setup was to compare gene expressions between apolipoprotein AI (apo AI) knockout mice and a reference of C57Bl/6 mice. Two main experiments were conducted. The first one compared eight different apo AI knockout mice with a reference pool of eight different C57Bl/6 mice on eight cDNA microarray slides. The second experiment compared the individual C57Bl/6 mice with the pool of them (same pool as above). In total 16 slides were hybridized. In this paper, we only present results based on the eight slides from the second experiment, although the same analysis on the eight apo AI slides gives similar results (indeed almost identical numbers). However, the first set of slides is believed to contain fewer differentially expressed genes compared to the other set. Hence, it is easier to argue that the underlying assumptions used in this analysis are true.

To further summaries the work by [4], part of the extracted RNA from the livers of the eight individual mice in each slide set was synthesized separately into cDNA together with Cy3-dUTP. A mix of the same extracted RNA was then used to produce the Cy5-dUTP labeled reference cDNA. A similar setup was used for the six unique apo AI knockout mice. For the production of the microarrays, approximately 5600 expressed sequence tags (ESTs) from the I.M.A.G.E. consortium database [7] were selected. 257 of these ESTs represent genes with possible and confirmed associations with lipid metabolism. The slides were spotted using a 4-by-4 print-tip arrayer with a total of 7056 spots on each slide. After hybridization the slides were scanned and the foreground and background signals were extracted using the Spot software [10]. Of the 21 rows in each print-tip group the last two were excluded resulting in a total of 399 spots per print-tip group or in total 6384 spots per slide. Among these spots, there were in total 5357 different genes and 840 blank spots. Among the 5357 genes, six of them were spotted trice, 175 of them duplicated and the rest were spotted only once per slide. In Figure 1, the

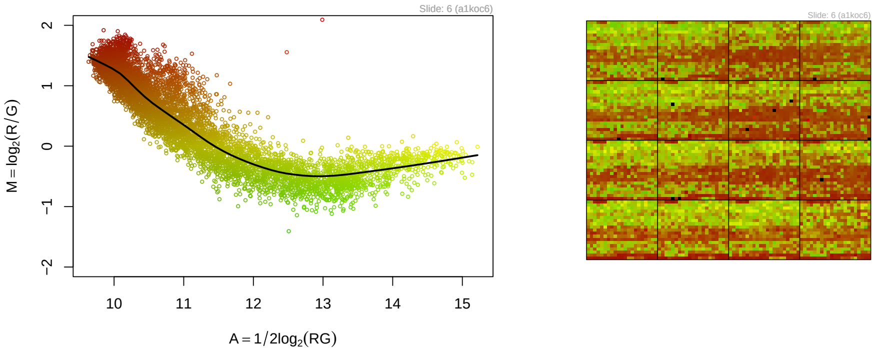


Figure 1: The log-ratios, M, of gene expression for the Cy3 and the Cy5 channels of the 6384 spots on slide 6. *Left*: The log-ratios, M, versus the log-intensities, A. The solid line is the fitted robust local regression line (lowess) and it shows the intensity-dependent bias effect of slide 6. *Right*: The same data plotted as laid out on the sixth cDNA microarray slide. Green colors correspond to a negative log-ratio and red colors to a positive log-ratio. The 4-by-4 grid separates the regions that were spotted by each individual print-tip.

log gene-expression ratios on slide 6 are shown. To the left, the logarithm of the gene-expression ratios, $M = \log_2(R/G)$, are plotted against the log-intensities, $A = 1/2 \cdot \log_2(R \cdot G)$, where in this case R and G are the non-background subtracted signals for the Cy5 (red) and Cy3 (green) channels, respectively. To the right, a spatial plot depicts the relative gene expressions as positioned on the slide.

3 Artifacts in data

As seen in the left plot in Figure 1, the log-ratios dependend strongly on the log-intensities. A low intensity spot tends to be redder (up regulated) than a high intensity spot, and this intensity dependency is non-linear. All slides show the same systematic effects. Due to the experimental setup, most genes are expected to be non-differentially expressed. For this reason, the observed non-linearity is an artifact and should be corrected for. There is a variety of methods available to correct for this effect, but, throughout this paper, we will rely on the intensity-dependent normalization methods suggested by [10]. Furthermore, investigating the spatial distribution of log-ratios, cf. the right plot in Figure 1, we find a repeated pattern of horizontal stripes for all slides in the data set. If we combine this plot with the M versus A plot and keep in mind that there are intensity-dependent artifacts, we conclude that these stripes may be due to different intensities between the stripes. The red areas with apparent up-regulated genes are darker, and the down-regulated and the non-differentially expressed genes tend to be located in brighter areas. We will return to this in the next section. As shown in Section 4.1, an intensity-dependent normalization removes most of the spatial artifacts, but our final findings indicate that the non-linearity between the Cy3 and Cy5 channels is not the only contributor to such systematic variations. Even if it would, it still has to be explained why the intensity varies spatially in such a regular pattern across the slides.

3.1 Systematic variation between plates

At each, so called, *dip*, the print-tips of the arrayer spot a number of (different) DNA clones onto the glass slide. In our case 16 spots are spotted at each dip. Depending on how the arrayer is programmed it either move on to the *rest of the slides* and spot the same set of clones (with possible intermediate refills of print-tips), or it retrieves a new set of clones from the microtiter plates (we say that the arrayer does a *source visit*) and finish the first glass slide before moving onto the next. The most commonly used approach is to spot all glass slides in the arrayer before spotting the next set of clones. The positions of all the spots printed at each dip are determined by the layout of the print-tips (here 4-by-4 print-tips approximately 200 μm apart) and the way the arrayer was programmed. Typically, the next set of clones (here 16) is spotted to the right of the ones from the previous source visit. This is repeated until one row is filled when the arrayer continues with the next row (starting over at the very left). Most modern arrayers allow user to spot the clones in any order and at any position, for instance column by column or randomly, but in most cases (also in this current setup) the arrayer moves from left to right row by row. With the knowledge of the exact printing of the slides one can create a plot where, say, the log-ratios of the 6384 spots are plotted in the order that the corresponding spots were printed onto the slide. A *print-order plot* is shown in Figure 2, with the log-ratio on the y-axis and the time of printing on the x-axis. From this plot it is clear that the log-ratios vary with the time point *when* the clones were spotted onto the slides. The systematic variation of the log-ratios over time is exactly that found to be repeated over the rows within each print-tip group, cf. right plot in Figure 1. The horizontal and vertical lines and the box plot to the right will be explained below. At a first glance it looks like there are

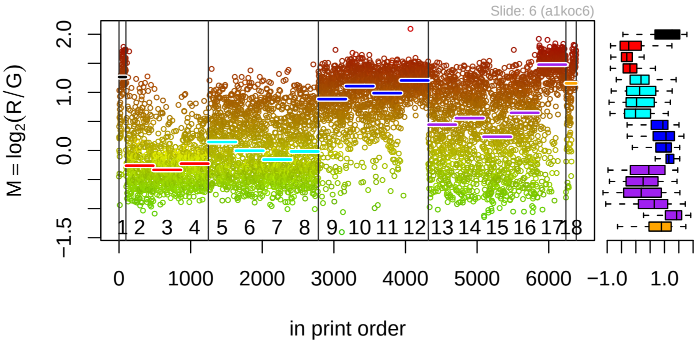


Figure 2: *Left*: The log-ratios of the 6384 spots on slide 6 in the order of when they were printed onto the slide. At each print dip, 16 spots were spotted simultaneously. The vertical lines mark the different clusters of plates. The horizontal lines are the biases of each plate group, which are colored according to which plate cluster (source) they belong to. *Right*: Box plot showing the median and the variability for each plate using the same ordering and coloring as in the left plot. Plate 1 is a the top and plate 18 at the bottom.

four to six printing periods (separated by vertical lines for clarity) with different properties. It turns out that the clones can be grouped into these groups based on what lab produced them. The slides used here were spotted using 18 different 384-well microtiter plates. Each of the vertical lines coincides exactly with a plate *and* lab switch. In Figure 2 the plate numbers are displayed along the x-axis. Thus, the graph may also be referred to as a *plate-order plot*. In order not to focus too much on different lab procedures, we refer to these different groups of plates as *plate clusters*. Plate 1 contains mostly of water (empty spots), plates 2-4 come from the I.M.A.G.E. clone library, plates 5-8 contain clones selected by the Kingley lab at Stanford University, plates 9-12 contain brain-tissue clones obtained by the Cheng Lab at Lawrence Berkeley National Laboratories, plates 13-17 were purchased from Research Genetics, and plate 18 comes from the Vulpe Lab. For plate 1, only 96 clones were spotted, and for plate 18, 144 clones were spotted. For all other plates all 384 clones were spotted. Furthermore, 95 out of the 96 spots (99%) from plate 1 are blanks, 336 out of 384 spots (88%) on plate 12, 326 out of 384 (85%) on plate 17, and 64 out of the 144 spots (44%) from plate 18 are blanks. In total there are 840 empty spots. In Figure 2 the plate clusters have been clarified by vertical lines. The horizontal lines, which are colored according to which plate cluster they belong to, are the *median log-ratios* for each plate. The box plot to the right depicts the distribution of the log-ratios for each plate. Plate 1 is at the top and plate 18 at the bottom. The same color scheme is used throughout the paper.

As suggested above, the systematic variations in data, especially the spatial effects seen in Figure 1 can be explained by a combination of systematic variation in the spot intensities across the slides and an intensity-dependent log-ratio effect. The print-order plot of the log-intensities in Figure 3 together with the print-order plot of the log-ratios in Figure 2 confirm this. Although differences in signal strengths (intensities) for different plates explain most of the spatial effects observed, the results show that there are additional plate effects of different origin.

3.2 Possible sources for different artifacts

The plate effects originate from the clones and the plates, that is, from a step before spotting the slides. In this sense, all slides should show approximately the same plate effects. The intensity-dependent effects, however, results from a non-linearity between the Cy3 and the Cy5 channels. There are at least two different possible sources for this non-linearity. First, the incorporation rate vary with the fluorescent dye used. For instance, the incorporation rate of Cy5-dUTP is about 60% (in moles) of that of Cy3-dUTP [5]. The second source of bias is introduced when the slides are scanned. Due to *quenching*, the Cy3 and the Cy5 dyes

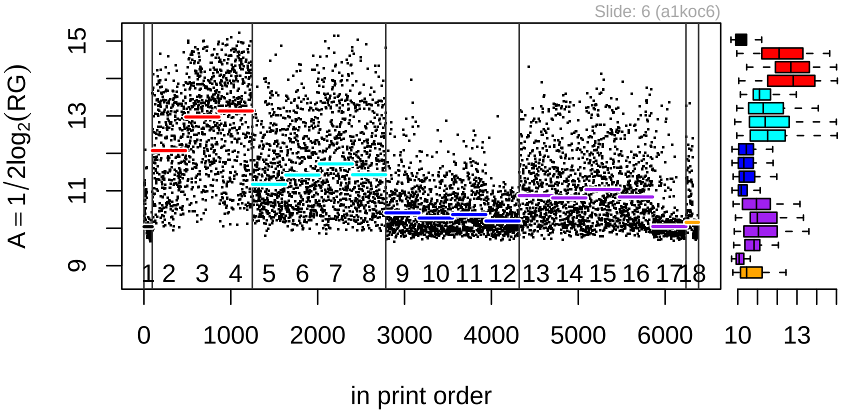


Figure 3: The log-intensities on slide 6. The plates with a lot of blanks (1,12,17 & 18) do have low intensities. Furthermore, the clones from brain tissue on plate 9-12 are, as expected, not very responsive to the test and control samples, which are from liver RNA. *Left:* The log-intensities in print order. *Right*: Box plot displaying the median and the variability of the log-intensities for each plate using the same ordering and coloring as in the left Figure.

are not linearly responsive over the full intensity range [8]. Since quenching is added after plate and labeling effects, it should be corrected for before the latter. However, because quenching and labeling effects show similar symptoms, it is hard to distinguish between them. Even if it is possible to identify the exact amount of quenching, a remaining problem is to separate the plate effect from the labeling effect. This is especially hard since these two effects are most likely combined in the hybridization step in an unknown way. We will attack the problem by assuming that the amount of quenching is not significant enought to be considered. Since we cannot distinguish between plate and labeling effects, we will test different sequences of normalization methods, referred to as *normalization strategies*, and use a measure of reproducibility (Section 5.1) to find the best one. This approach will guide us to which artifact is the most dominant one.

4 Normalizing data

Plate effects can be taken into account in many different ways when data is normalized. One may either search for a physical model to explain the effects, or use a blindfolded black-box model to normalize the observed signals. Here we use the latter approach, because we do not know in what step(s) or combination of steps of the cDNA microarray process plate effects are introduced, but also because it is a more objective approach for arguing for existence of plate-dependent effects. Next, we will present a few alternatives to normalize plate effects, and in Section 5.1 we define a *measure of reproducibility* in order to compare them.

4.1 Intensity-dependent normalization

Figure 4 shows the log-ratios after scaled print-tip intensity normalization [10]. Spatial artifacts are less significant after the normalization, but the variation between the plates are still there. A possible strategy is to remove the plate biases after the intensity normalization, cf. Section 4.4. For details on intensity-dependent normalization methods, see [10].

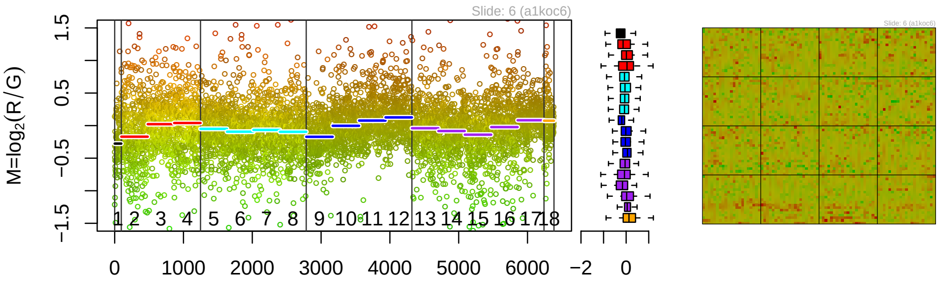


Figure 4: Gene expressions (for slide 6) *after* scaled print-tip intensity normalization. *Left*: Print-order plot of the log-ratios with a horizontal box plot showing the distribution of the log-ratios within each plate group. *Right*: The spatial location of the log-ratios. Normalizing for intensity-dependent artifacts removes much of the spatial and some of the plate effects.

4.2 Platewise median normalization

An alternative way to normalize data is found if we assume that the average log-ratio on each plate or plate cluster is zero. This is similar to the assumption used for normalization where one assumes zero shift, either for all spots or for each print-tip group. These two assumptions look similar, but there is an intrinsic difference between them. The clones on the plates might be selected in such a way that some plates may contain *many* clones that are differentially expressed whereas such an *a priori* selection of clones is less likely to affect the print-tip groups, because they contain clones from *all* plates and are therefore more random. The assumption of zero shift for each plate is violated if the expression levels of the clones on a plate differ significantly between test and control samples. For instance, this may be the case with the apo AI versus control pool data set. We do not believe this is true for the self-to-self mouse experiment investigated here, because it is based on eight control mice versus the pool of them. Hence, we assume that all plates should have an average log-ratio of zero, which satisfies the assumption that the overall shift on the slide is zero. In Figure 2, the median log-ratio of each plate is depicted by horizontal lines. The result from normalizing data plate by plate assuming zero bias, that is, by shifting all the horizontal lines in Figure 2 to zero, is illustrated in the M versus A plot and the spatial plot in Figure 5. Comparing these with the corresponding plots for the non-normalized

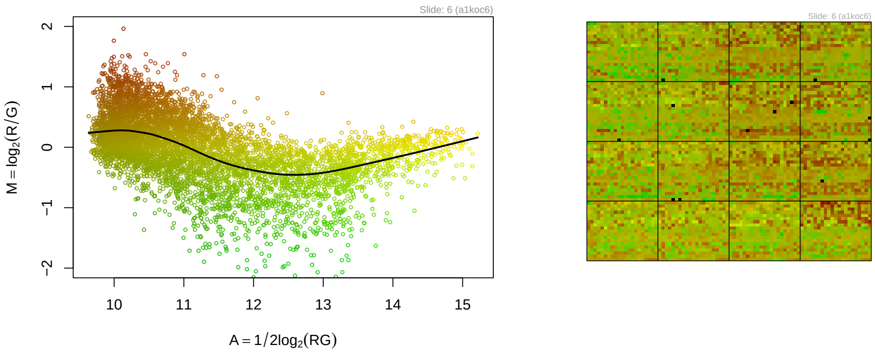


Figure 5: Gene expressions (for slide 6) *after* platewise bias normalization. *Left*: The log-ratios versus the log-intensities. *Right*: The spatial location of the log-ratios. Removing the constant bias within each plate group removes a lot of the spatial and some of the intensity-dependent effects.

data (Figure 1) we find that intensity-dependent effects are partly removed by plate-bias normalization and that spatial effects are less significant. Remaining intensity-dependent biases are studied in Section 4.4.

4.3 Intensity-dependent normalization performed platewise

An alternative strategy is to apply intensity normalization for each microtiter plate individually. In Figure 6 the lowess-estimated intensity curve is shown for each plate. In this case we have to be careful, because the number of data points might

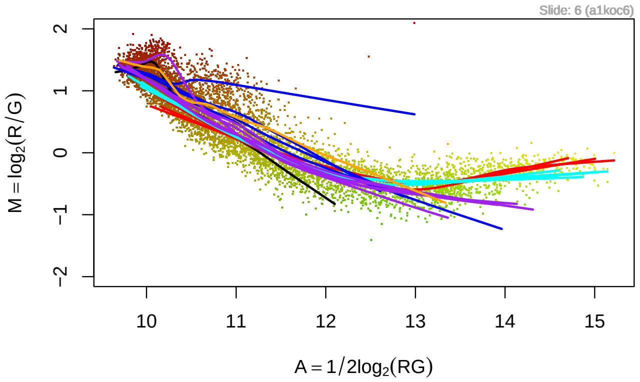


Figure 6: Intensity dependency for each plate (on slide 6). The top most curve (blue) of plate 12 is shifted upward because it has only one data point in the upper intensity range, $A = [12, 13]$, with a high value of the log-ratio.

not be large enough in order for robust smoothing estimates to be robust over the full intensity range. In such cases we have to look for larger groups, see Section 4.6, or find a clever way to combine different fitted curves, cf. [9]. This may be an issue for plate 1 and plate 18, but for simplicity we ignore such problems in this study. We do this because we do not believe it affects our findings in general. However, if we were searching for, say, differentially expressed genes, we would have to consider this problem more carefully. The platewise intensity normalized data for slide 6 is depicted in the print-order and spatial plots in Figure 7.

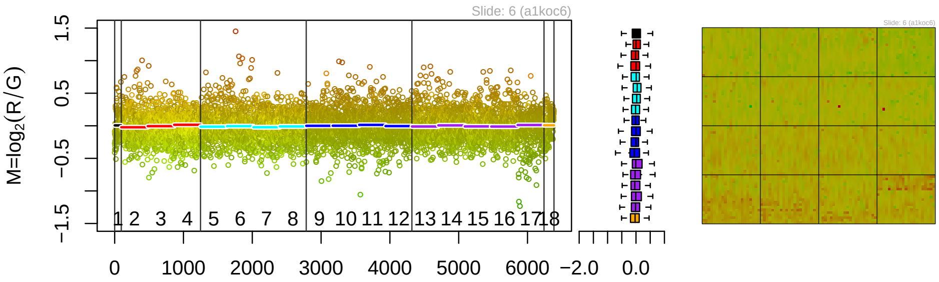


Figure 7: Gene expressions (for slide 6) *after* platewise intensity-dependent normalization. *Left*: Print-order plot of the log-ratios with a horizontal box plot show the distribution of the log-ratios within each plate group. *Right*: The spatial location of the log-ratios.

4.4 Subsequential normalization strategies

When a platewise constant-shift normalization is applied some of the intensity effects are removed, but not completely as seen in Figure 5. Remaining effects can be removed by utilizing standard intensity normalization. However, care has to be taken since this will correct the plate-normalized signals *too much* and reintroduce "new" plate effects. A naive approach is to do another round of plate-dependent bias normalization. A similar strategy can be used to correct for remaining plate effects after an initial intensity-dependent normalization, or any other type of normalization. More importantly, the existance of both plate *and* intensity effects raises the question of which one of the two effects is the most dominant one, and which one should be corrected for first, or if both effects should be corrected for simultaneously. We leave the problem of simultaneous normalization to the future and focus on the simpler case, which is most likely the one that will be used in practice. Recall Section 3.2 for a short discussion on this problem.

4.5 About scale normalization

Since some plates have clones from different tissues and have been prepared by different labs and people, it is likely that their variabilities differ. For this reason, we do *not* assume that all plates have equal variability, and therefore we do no consider scale normalization of each plate group. For the same reason, we do not assume equal average *intensity* across plates. In addition, the noise and signal levels in the two channels differ between plates, and therefore the variances of the log-ratios as well as log-intensities are expected to differ between plates. However, it *is* possible to assume equal variability across print-tip groups, because of "semi-random" positioning of clones. It is also reasonable to assume log-ratios to be zero on average.

4.6 Alternative ways to group data

Instead of focusing on plates, we may define other groups that are normalized separately. The main thing to keep in mind is, as for almost all normalization methods, that we must be confident that it is possible to assume that the average log-ratio of group is zero. For instance, in our case we could have grouped data into one group containing all the empty spots, and the rest of the clones grouped according to in which lab they were created. Another possibility is to estimate the average plate median shifts (or the plate intensity-dependent shifts) for all (standardized) slides, and use these platewise constants (or smooth curves) to normalize each slide individual. However, in this paper, we only consider platewise normalizations.

5 Comparison of different strategies

We compare seven classes of normalization strategies to each other and to the non-normalized case. In one group we normalize data for (slide, print-tip and scaled print-tip) intensity-dependent effects (labeled Sl(A), Pr(A) and sPr(A) in tables and figures). In a second group we normalize data for (constant) plate-dependent effects (labeled Pl), and in a third group we apply intensity-dependent normalization to each plate group (Pl(A)). In addition, we test different order of normalization sequences for each of them. In one group we first unshift the plate biases and then do standard intensity normalizations (labeled Pl-Sl(A), Pl-Pr(A) and Pl-sPr(A)). The reverse order is also considered (labeled Sl(A)-Pl, Pr(A)-Pl and sPr(A)-Pl). Furthermore, we do plate-intensity-plate normalization (Pl-Sl(A)-Pl, Pl-Pr(A)-Pl and Pl-sPr(A)-Pl). Finally, we combine intensity normalization over plates with standard intensity normalization methods, in that order (Pl(A)-Sl(A), Pl(A)-Pr(A) and Pl(A)-sPr(A)), and in the reverse order (Sl(A)-Pl(A), Pr(A)-Pl(A) and sPr(A)-Pl(A)). Each of the above is applied to both background (labeled bg) and non-background subtracted signals. In total, we compare $2 \cdot (1+3+1+1+3+3+3+3+3) = 42$ different strategies. For each of the strategies, we rescale the log-ratios so that all slides have the same MAD (within each normalization strategy, *not* between). More precisely,

$$Z_l \leftarrow \text{MAD}_{k=1:K} M_{k,l},\ \forall l$$

$$\bar{Z} \leftarrow \left(\prod_{l=1}^{L} Z_l \right)^{1/L}$$

$$M_{k,l} \leftarrow \frac{\bar{Z}}{Z_l} \cdot M_{k,l},\ \forall k, l$$

where $k = 1, \ldots, K$, $l = 1, \ldots, L$, K is the number of spots on each slide, and L is the number of slides (not to be confused with the number of genes and the number of replicates).

5.1 Measure of reproducibility

In order to identify an optimal *normalization strategy*, we measure the *variability of the gene-by-gene residuals*. The average variability and the variability of the variabilities can be seen as a measure of reproducibility. Ignoring the bias, the smaller the variabilities are, the more reproducible data is. Since we do not know the true values of the expression ratios, we cannot test if the different normalization methods introduce bias. For this reason, we cannot give rigorous evidence that normalized data is more *consistent*. However, all normalization methods described above (except the scale normalization) use additive operators, not multiplicative. Therefore, we believe the risk for overfitting data is low. Furthermore, if we search for the most differentially expressed genes among several, a small bias in the average expression ratio has little impact on the identified genes.

The measure of reproducibility is defined as follows. Let d_i be the variability of gene i calculated as the median absolute deviation of the gene residuals:

$$d_i = 1.4826 \cdot \operatorname*{median}_{j=1:J} |r_{i,j}|,$$

where 1.4826 is a constant to make MAD a consistent estimator of the standard deviation for normally distributed data. The residuals are defined as

$$r_{i,j} = M_{i,j} - \operatorname*{median}_{j=1:J} M_{i,j}$$

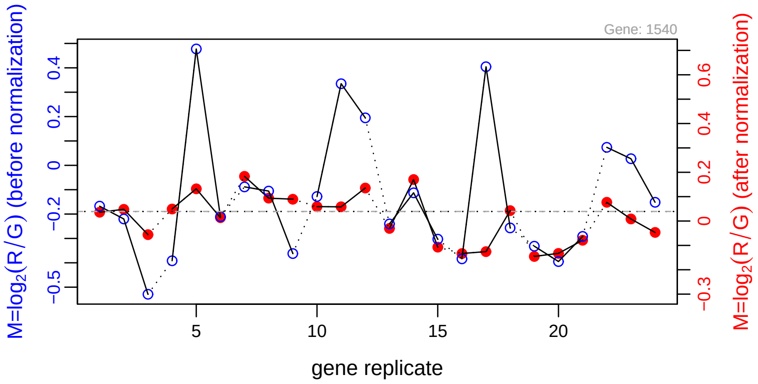


Figure 8: The log-ratios for all the 24 replicates of gene 1540 found on the eight slides after first performing intensity normalization for each print-tip group and then for each plate group (red discs), compared to not doing normalization at all (blue circles). The left scale is for normalized log-ratios and the right one for non-normalized ones. Note that the unit lengths of the two vertical axes are equal, but their origins have been aligned to simplify visual comparison.

where $M_{i,j}$ is the log-ratios of gene i and replicate j. In Figure 8 the log-ratios for the 24 replicates of the triplicated gene 1540 are plotted for two different strategies. For this specific gene, print-tip followed by plate intensity normalization (red discs) makes the measurements (of this gene) *more similar* compared to the non-normalized approach (blue circles). The former has a genewise MAD of 0.115 (sample standard deviation is 0.0994) whereas the latter has a MAD of 0.190 (0.264). Obviously, this may only be true for this specific gene. Therefore, we look at the *overall distribution of the gene-by-gene variabilities*. The latter can be summarized by, say, the 95% quantile, or the mean, of the genewise variabilities $\{d_i\}_i$. The mean is not only a measure of the most common variability, but it is also sensitive to genes with high variabilities;

$$MOR. = \frac{1}{N}\sum_{i=1}^{N} d_i$$

where N is the number of genes. Note that, the *smaller* MOR is the *more* reproducible data is. In general, rescaling log-ratios across all slides increases (worsen) the MOR. However, rescaling is commonly small and its effect on the MOR and, hence, on the final findings is therefore of subordinary order.

6 Results

Our investigations show that it is optimal to normalize the data set in hand for intensity-dependent effects both within each print-tip group (Pr(A) or sPr(A)) and within each plate group (Pl(A)). The overall optimal strategy, for this data set, is to start with a scaled print-tip intensity normalization (sPr(A)) *followed* by a platewise intensity normalization (Pl(A)). The first one reduces the genewise variability by approximately 55%, and the second one reduces it another 5% relative to the variability of the non-normalized data (or approximately 10% compared to the proceeding one). See Table 1 and the box plot in Figure 9. This indicates that, in addition to intensity-dependent effects, there exist *plate-dependent effects*. Moreover, we find that it is always better to apply intensity-dependent normalization to each print-tip group individually (Pr(A) or sPr(A)) than to apply it to all spots at once (Sl(A)). We believe this is because normalization print-tip by print-tip corrects for spatial effects, which the latter is not capable of.

We find that *not* doing background correction give in more coherent measurements across slides and across replicates. This was true all normalization strategies tested. The only case when background subtraction gave less variability was when no normalization was performed (labeled 'none' in the table and in the box plot).

Almost identical results are obtained if the sample standard deviation is used instead of the sample median absolute deviation in the calculation of the variability of the gene residuals.

Finally, performing the same analysis on the apo AI knockout mice data set, which is expected to have a few differentially expressed genes, also gives that a print-tip followed by a plate intensity normalization performs better than a print-tip intensity normalization alone. For the same data set, the authors of [10] identify eight genes (or ESTs) to be the most differentially expressed genes. The top ten genes (the eight listed in their paper plus two others) were confirmed by real-time quantitative polymerase chain reaction (QRT-PCR) to be differentially expressed. All normalization strategies that we found to be the best ones do all rank these ten genes as the most differentially expressed genes. However, these genes are so

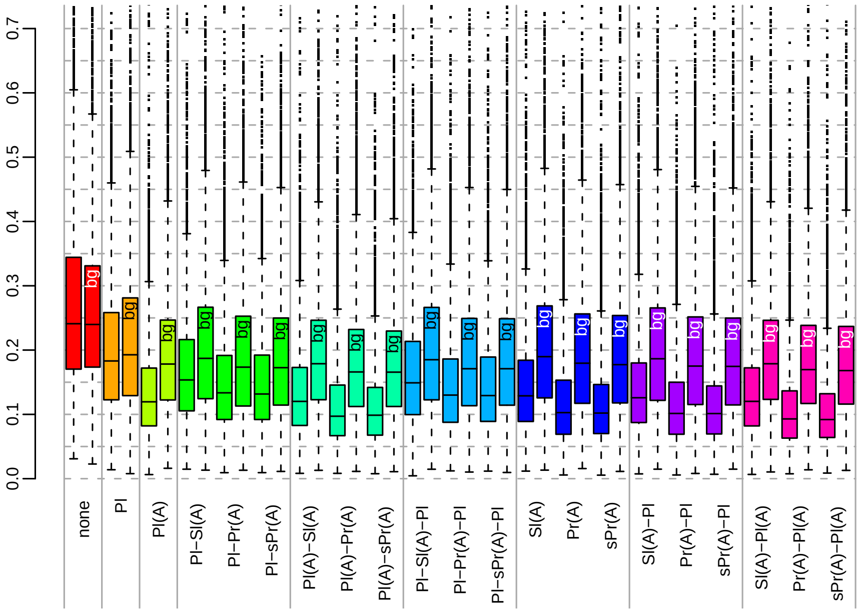


Figure 9: Comparison of sample distributions of the gene log-ratio variabilities after (2 × 21) different normalization strategies have been applied. Note that the tails of the sample distributions have been cut of at 0.75 and that they do extend (approximately equally far) beyond this level. Legend: none - No normalization, Pl - Constant plate bias removed, Pl(A) - Intensity-dependent plate bias removed, Pr(A) - Print-tip intensity-dependent bias removed, sPr(A) - Scaled print-tip intensity-dependent bias removed, Sl(A) - Slide intensity-dependent bias removed. Normalization strategies that start out with background subtracted signals are, in addition, labeled with 'bg'. For example, Pl-sPr(A) refers to a normalization sequence with a plate normalization followed by a scaled print-tip normalization.

extreme that almost any normalization strategy applied will identify them to be the most differentially expressed ones. Hence, we may only conclude that our best methods will not remove the signal for these top ten genes.

7 Discussion

The actual source of the plate effects is unknown, but one reason may be that there are different concentrations in the spots, which in turn may be due to different levels of evaporation for different plates.

The large difference between performing background correction and not is interesting. The subtraction of background from the foreground signals introduces noise, and the effect is enhanced by the log-ratio log-intensity transform. However, we do not believe it is of the same order of magnitude as observed for almost all of the normalization strategies tested. We believe that the background estimates, which are based on measurements from the surrounding regions of the spots, are not the same as the background noise that affects the internals of the spots. A physical explanation for this may be that the background noise found in the proximity of the spots do not stick to the spots themselves, simply because where the spots are the glass is already covered by cDNA and nothing but matching cDNA sequences can hybridize to the foreground areas. Furthermore, some (not all) of the background estimates from the Spot image analysis software show clear print-order effects too. This was surprising and has to be investigated further.

Finally, an argument against our measure of reproducibility is that it is only including the variance. The reason for this is that we do not know the true gene-expression levels for any genes. However, recently it has become more common to add control spots to cDNA microarray slides such as spike-ins, negative and positive controls etc. We are currently working on methods for comparing normalization strategies using both the bias and variance information of such control spots.

Acknowledgments

I am in big debt to professor Terry Speed at the Statistics Department, University of California, Berkeley, for supervising me while I was at the Statistics Department at UC Berkeley in 2000/2001. I am very thankful to Jean Yang and the rest of the Speed Group for taking the time to explain the cDNA microarray technique and for fruitful discussions. I would also like to thank Matt Callow at the

Lawrence Berkeley National Laboratory for providing data and answering all my questions about details in the experimental setup. Also, if it would not be for the discussion that I had with Karen Berger at the Monica Moore's Lab, Ernest Gallo Clinic & Research Center (Emeryville) in July 2002, I probably would not have come up with the print-order plot in the first place. I would also like to thank my supervisors Ola Hössjer and Jan Holst for feedback and proof reading. Finally, thanks to all the contributors to the R project, which provides me with an important foundation for applied research.

This work was supported by the following grants: The Swedish Foundation for International Cooperation in Research and Higher Education (STINT), The Foundation BLANCEFLOR Boncompagni-Ludovisi, née Bildt, The Fulbright Commission Grant, The Ernhold Lundströms Stiftelse and The Royal Swedish Academy of Sciences Research Grant.

Method	0%	1%	50%	MOR	99%	100%	%MOR
none	0.0309	0.0615	0.241	0.270	0.708	1.17	1
bg-none	0.0228	0.0631	0.240	0.264	0.676	1.23	0.977
Pl	0.0139	0.0375	0.183	0.204	0.588	1.59	0.756
bg-Pl	0.0078	0.0393	0.193	0.218	0.623	1.08	0.806
Pl(A)	0.00637	0.0261	0.120	0.139	0.438	1.22	0.514
bg-Pl(A)	0.0161	0.0440	0.178	0.199	0.597	1.11	0.739
Pl-Sl(A)	0.0147	0.0389	0.154	0.173	0.515	1.36	0.641
bg-Pl-Sl(A)	0.0133	0.0401	0.187	0.209	0.600	0.929	0.776
Pl-Pr(A)	0.00933	0.0299	0.134	0.154	0.495	1.29	0.569
bg-Pl-Pr(A)	0.0131	0.0330	0.173	0.196	0.588	0.953	0.726
Pl-sPr(A)	0.00939	0.0285	0.132	0.154	0.500	1.34	0.570
bg-Pl-sPr(A)	0.0113	0.0329	0.173	0.195	0.595	1.04	0.724
Pl(A)-Sl(A)	0.00829	0.0252	0.120	0.139	0.440	1.22	0.516
bg-Pl(A)-Sl(A)	0.0130	0.0443	0.179	0.200	0.598	1.11	0.740
Pl(A)-Pr(A)	0.00827	0.0223	0.0971	0.119	0.428	1.17	**0.441**
bg-Pl(A)-Pr(A)	0.0113	0.0362	0.166	0.185	0.575	1.00	0.686
Pl(A)-sPr(A)	0.00771	0.0231	0.0987	0.118	0.445	1.41	**0.438**
bg-Pl(A)-sPr(A)	0.0109	0.0370	0.166	0.185	0.576	1.21	0.684
Pl-Sl(A)-Pl	0.00444	0.0353	0.149	0.168	0.503	1.08	0.621
bg-Pl-Sl(A)-Pl	0.0147	0.0382	0.185	0.208	0.594	0.940	0.769
Pl-Pr(A)-Pl	0.0121	0.0273	0.130	0.149	0.479	1.09	0.553
bg-Pl-Pr(A)-Pl	0.0103	0.0313	0.171	0.194	0.583	0.988	0.718
Pl-sPr(A)-Pl	0.0117	0.0279	0.129	0.150	0.486	1.06	0.556
bg-Pl-sPr(A)-Pl	0.00964	0.0320	0.171	0.194	0.599	1.08	0.718
Sl(A)	0.0118	0.0310	0.129	0.149	0.499	0.907	0.552
bg-Sl(A)	0.0133	0.0417	0.190	0.212	0.631	1.00	0.786
Pr(A)	0.00575	0.0234	0.103	0.124	0.446	0.924	0.460
bg-Pr(A)	0.0157	0.0349	0.180	0.200	0.592	1.05	0.741
sPr(A)	0.00549	0.0228	0.102	0.123	0.480	1.27	0.455
bg-sPr(A)	0.0113	0.0349	0.177	0.199	0.601	1.26	0.737
Sl(A)-Pl	0.00746	0.0297	0.126	0.147	0.499	0.900	0.543
bg-Sl(A)-Pl	0.0148	0.0365	0.186	0.209	0.618	0.944	0.775
Pr(A)-Pl	0.00575	0.0220	0.101	0.122	0.446	0.911	0.453
bg-Pr(A)-Pl	0.0081	0.0331	0.175	0.196	0.587	0.977	0.727
sPr(A)-Pl	0.00682	0.0231	0.101	0.121	0.476	1.26	0.449
bg-sPr(A)-Pl	0.0148	0.0348	0.175	0.196	0.606	1.18	0.726
Sl(A)-Pl(A)	0.00661	0.0253	0.120	0.139	0.447	0.983	0.514
bg-Sl(A)-Pl(A)	0.0104	0.0436	0.179	0.199	0.599	1.12	0.739
Pr(A)-Pl(A)	0.0076	0.0217	0.0929	0.111	0.394	0.848	**0.412**
bg-Pr(A)-Pl(A)	0.0138	0.0369	0.170	0.189	0.553	0.950	0.702
sPr(A)-Pl(A)	0.00872	0.0220	0.0919	0.110	0.424	1.01	**0.407**
bg-sPr(A)-Pl(A)	0.0129	0.0372	0.168	0.188	0.562	1.15	0.696

Table 1: Comparison of different normalization strategies. Legend: none - No normalization, Pl - Constant plate bias removed, Pl(A) - Intensity-dependent plate bias removed, Pr(A) - Print-tip intensity-dependent bias removed, sPr(A) - Scaled print-tip intensity-dependent bias removed, Sl(A) - Slide intensity-dependent bias removed. Normalization strategies that start out with background subtracted signals are, in addition, labeled with 'bg'.

References

[1] Henrik Bengtsson. com.braju.sma - object-oriented microarray analysis in 100% R. http://www.maths.lth.se/help/R/, 2001.

[2] Henrik Bengtsson. R.classes - a bundle for object-oriented programming with reference in R. http://www.braju.com/R/, 2001. Mathematical Statistics, Centre for Mathematical Sciences, Lund University, Sweden.

[3] Ben Bolstad, Sandrine Dudoit, Ingrid Lönnstedt, Natalie Roberts, and Jean Yee Hwa Yang. R package: Statistics for Microarray Analysis (sma). http://www.stat.berkeley.edu/users/terry/zarray/Software/smacode.html, 2001.

[4] M. Callow, S. Dudoit, E. Gong, T. Speed, and E. Rubin. Microarray expression profiling identifies genes with altered expression in HDL-deficient mice. *Genome Research*, 10(12):2022–9, December 2000.

[5] Priti Hegde, Rong Qi, Kristie Abernathy, Cheryl Gay, Sonia Dharap, Renee Gaspard, Julie Earle-Hughes, Erik Snesrud, Norman Lee, and John Quackenbush. A concise guide to cDNA microarray analysis. *Biotechniques*, 3(29):548–562, September 2000.

[6] Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

[7] Greg Lennon, Charles Auffray, Mihael Polymeropoulos, and Marcelo Bento Soares. The I.M.A.G.E. Consortium: An integrated molecular analysis of genomes and their expression. *Genomics*, 33:151–152, April 1996.

[8] Latha Ramdas, Kevin R. Coombes, Keith Baggerly, Lynne Abruzzo, W. Edward Highsmith, Tammy Krogmann, Stanley R. Hamilton, and Wei Zhang. Sources of nonlinearity in cDNA microarray expression measurements. *Genome Biology*, 2(11):research0047.1–0047.7, 2001.

[9] Yee Hwa Yang, Sandrine Dudoit, Percy Luu, David M. Lin, Vivian Peng, John Ngai, and Terence P. Speed. Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucelic Acids Research*, 30(4):e15, Feb 2002.

[10] Yee Hwa Yang, Sandrine Dudoit, Percy Luu, and Terence P. Speed. Normalization for cDNA microarray data. Technical Report 589, Department of Statistics, University of California at Berkeley, 2000.

C

Paper C

The R.oo package - Object-oriented programming with references using standard R code

Henrik Bengtsson

Mathematical Statistics at the Centre for Mathematical Sciences, Lund University, Box 118,
SE-221 00 Lund, Sweden. Email: hb@maths.lth.se.

ABSTRACT

When designing and implementing object-oriented applications in R, problems concerning generic functions and reference variables often occur. It is currently not clear how to create generic functions in a robust way such that a new package will be compatible with existing or future packages. This becomes an important issue as more and more packages are made available. As a functional language, R does not provide methods for programming with references as all arguments to functions are copied by value. However, there are situations where it is useful or even necessary to pass arguments by references. In this paper we present the package R.oo, which overcomes these problems. In addition to a way for automatically creating generic functions in the background and a transparent way of using references, it also provides a foundation for designing and implementing object-oriented applications in a robust way. In this context, we also suggest a draft of a coding convention intended to bring additional structure to the source code. The package also extends the current exception handling mechanism in R such that exception objects can be caught based on their class.

Keywords: Object-oriented programming, generic functions, reference variables, coding conventions, exception handling, root class, S3, UseMethod.

1 Introduction

In R there are currently two ways for programming with classes, which are commonly referred to as the S3 (or S3/UseMethod) style [12] and the S4 style [6].

The S3 style has been supported by R from the very beginning and support for the S4 style was added with the release of the *methods* package [7], which became part of the core distribution as of R v1.4.0 and is loaded by default from R v1.7.0. The package presented in this paper is currently relying exclusively on the S3 programming style, meaning that all classes defined using the package become S3 classes. However, future versions are likely to migrate to the S4 programming style. Having said this, it should be clear that the intention of the R.oo package is not to replace S3 or S4, but to extend them with a layer such that object-oriented design and programming in R become easier and more robust.

For beginners, but also for experts, it can be quite tedious to implement an object-oriented design in a robust way using the standard S3 (or S4) schema. It is easy to forget to add a generic function or, worse, to overwrite a non-generic function by a generic function. As a developer one has to stay up to date with all generic functions and all functions implemented in the latest R distribution to be sure that core functionalities in R are *not* overridden or removed when one's package is loaded. To be certain that the package works anywhere one must stay up to date with all other packages that the end user might load in parallel with your package. This approach is not scalable as more and more packages are added to the CRAN.[7]

In R arguments are supplied to functions by a pass-by-value semantic, also known as call-by-value. Inside the function, the arguments behave like local variables and any change made to a value inside the function is not reflected in the original value. The rational for this is that a function by definition takes one or several input values, returns something else and it should never modify the input values. When passing large data objects to a function a true pass-by-value approach would be too expensive in terms of time and memory. To overcome this, an argument in R is in practice *passed by reference* as long as the variable is not modified by the function. As soon as the function changes the value of the argument, a local copy of it is created to obtain *pass by value*. Occasionally there are requests for adding a true pass-by-reference semantic so that functions *can* modify the original variable. From now on we refer to such functions as methods and reserve the word *function* for its original meaning. There are different ways to emulate a pass-by-reference

[7] The current effort of adding *namespaces* [11] to the R language will remove some of the problems related to generic functions. A possible future support for *multiple generic function* will most likely solve the problem completely.

semantic by letting regular R variables represent *reference variables* or shortly *references*. With references it is possible to write more memory and time efficient code. Another advantage is that it is possible to write more user-friendly methods where fewer arguments need to be specified by the end user.

For reasons like these the R.oo package was developed. Its purpose is to relieve the developer from implementation details to make it possible to focus on the object-oriented design. The utility functions *setConstructorS3()* and *setMethodS3()* create constructor functions and class specific methods and at the same time make sure that required generic functions are created (or not) and throw exceptions if illegal method names are used, e.g. reserved words. Furthermore, the package provides a totally transparent way of using *references* by defining a new "data type", namely the class *Object*. Any object of a class that inherits from this root class will be passed to functions as a reference. Using a single root class, which all classes inherit from, improves the overall design and structure of any package developed. Finally, the R.oo package also provides an extended exception handling mechanism where an exception can be thrown and then caught depending on its class.

The R.oo package was written using standard R code ("100% R") and runs on all platforms. It was designed and implemented in an object-oriented style.

This document will *not* discuss what object-oriented programming and design are about. For an extensive case study on how to use the R.oo package see [2]. In Section 2, we describe how to use classes that are defined using the R.oo package and that inherits from the root class *Object*. In this section some additional object-oriented features that come with the R.oo package are also described. To further improve the structure of an object-oriented implementation, we suggest an R Coding Convention (RCC) in Section 3. The utility functions for defining constructors and methods, which are introduced in Section 4, assert that the RCC is followed. The root class named *Object* is described in detail in Section 5, and the usage of references, which is provided by the *Object* class, is discussed in Section 6. Section 7 is about exception handling and Section 8 describes some other utility functions that is part of the package. Other classes defined in the package are briefly mentioned in Section 9. Section 10 explains how to download and install the package. Conclusions are given in Section 11.

2 Using classes

Throughout this document we make use of a simple case study example to describe the major parts of the package. Let the class SavingsAccount represent a bank account, which in the simplest case can be described by its balance. To secure against illegal modifications of the balance we represent the balance with a private field named `.balance` (see Section 3). To obtain the balance of the account the function *getBalance()* is provided. Using *setBalance()* it is possible to modify the account balance directly, but it is not possible to set it to a negative balance. More commonly used are the methods for withdrawal and depositing, i.e. *withdraw(amount)* and *deposit(amount)*, respectively. The withdrawal method will not accept withdrawals if the balance becomes negative. The class inherits from the root class *Object* (Section 5). When a SavingsAccount object is created, the balance will by default be set to zero. A Unified Modeling Language (UML) model of the SavingsAccount class is depicted in Figure 1.

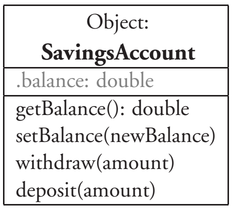


Figure 1: UML representation of the SavingsAccount class, which extends the *Object* class. Private fields have a . (period) as a prefix. 'Object:' in the header means that the class extends the *Object* class.

2.1 Creating an object

The implementation of the class is described in Section 4, but for now assume that the usage of the constructor is `SavingsAccount(balance=0)` and that

```
account <- SavingsAccount(100)
```

creates a SavingsAccount object with initial balance 100. The object is referred to by the reference variable `account`.

2.2 Accessing fields

The fields of an instance of class inheriting from root class *Object*, shortly *an Object*, can be accessed similar to how elements of a list are accessed. For example, the balance field of the account object can be retrieved by either `account$.balance` or `account[[".balance"]]`. To set the balance of the account either `account $.balance <- newBalance` or `account[[".balance"]] <- newBalance` will do.

Note that there is no way to prevent the access to *private* fields. However, if one follows the RCC rule [3] that private fields and only private fields should have a `.` (period) prefix, it should be clear which fields should be accessed from outside and which should only be accessed from inside the SavingsAccount class. Moreover, private fields named this way will, by default, not be listed by the functions *getFields()* and *ll()*, which are described further in Section 5, and neither by *ls()* [10].

2.3 Calling methods coupled with a class

Under the S3 schema a method coupled with a class is called in the same way as a regular function, but with (the reference to) the object as the first argument. When specifying the withdrawal and depositing methods above, we excluded the object argument for simplicity, e.g. *withdraw(amount)*. However, when calling the method one has to include it, e.g. `withdraw(account, amount)`. The method dispatching mechanism in S3/UseMethod then makes sure that the method of the correct class is called. For detailed information on how method dispatching is done in S3, see [10].

Similar to how fields are accessed, methods that are coupled with any *Object* derived class, can be accessed via the `$` (or the `[[`) operator, e.g. `account$ withdraw(amount)`. In many other object-oriented languages such as Java and C++, but also the Omegahat's OOP project [8], this is the format used for methods calls. However, we do not recommend this style, except for static methods. Methods in R should be though of as belonging to generic functions [6] and not to a specific class per se.

2.4 Calling static methods

A *static method* of a class is invoked using only the class name and it does not require an instance of a class. All classes extending the *Object* class may define static methods. The most readable way to call a method of a class is via the `$` operator, e.g. `Object$load(file)` and `Exception$getLastException()`, even though `load(Object(), file)` and `getLastException(Exception())` are also possible.

2.5 Accessing virtual fields

For *Object* instances, there is a third way of calling methods. Methods with a name of format `get<`*`Field`*`>(object)` or `set<`*`Field`*`>(object, value)` can be accessed by what we denote as *virtual fields*, i.e. as `object$<`*`field`*`>`. For instance the methods `getBalance(account)` and `setBalance(account, newBalance)` will be called whenever `account$balance` and `account$balance <- value` are evaluated, respectively.[8]

There are at least three real advantages of using virtual fields. First, it is possible, as the name suggest, to make it look like a class has a certain field whereas it internally might use something else. For instance, a Circle class can have the two redundant fields named `radius` and `diameter` where one is a virtual field and the other is the actual field. Indeed, both might be declared virtual at the same time. We find that the use of virtual fields reduces the redundancy, which in turn reduces the risk for inconsistency. It also reduces the memory usage. Another advantage is that it is possible to restrict what values can be assigned to a field. For instance, we can prevent the user from setting a negative radius, e.g. `circle$radius <- -20`. Finally, virtual fields can prevent direct access to private fields or modification of constants, that is, they provide a mechanism for *encapsulation* (data hiding).

[8] By default, virtual fields have higher priority than regular fields, meaning that if a virtual field exists it will be accessed first, but it is possible on a reference-to-reference or an object-to-object basis to change the order which fields, virtual fields, methods and static methods are accessed.

2.6 Accessing class fields

A *class field*, also known as a static field, is a field associated with the class itself, not with a particular instance of the class. A class field of a class is shared by all objects of that class. A common role of a class field is that of a global variable (except from the important difference that it is not a global variable), e.g. `Colors$RED.HUE`. A class field is accessed as a regular field, except that the object is now the static class object, e.g. `SavingsAccount$count <- SavingsAccount$count + 1`. Any class extending the *Object* class can have static fields.

3 Coding conventions

An important part of object-oriented design and implementation is to follow a standard for describing the design and for implementing it. There are several standards for describing object-oriented design of software, but one that has become the major standard is the Unified Modelling Language (UML) [9]. For implementation standards, also referred to as *coding conventions*, some languages have a well defined specification to follow whereas others do not. Unfortunately, there is no explicit and official coding convention for R. A well defined coding convention is useful because it helps to make the code more structured and more readable, and it reduces the risk for mixing up field names with class names or reassign fields that are supposed to be constants etc. It is also fundamental in order to efficiently share source code between developers and over time. For this reason we are working on a *R Coding Convention* (RCC) draft [3]. Next we will present an excerpt of its naming conventions.

3.1 Naming conventions

Some of the naming convention rules of the RCC apply to object-oriented design and programming. One of the most important is how classes, fields and methods should be named. According to the RCC, names that represent classes must be nouns and written in mixed case starting with upper case, e.g. `SavingsAccount`. Both field and method names must be in mixed case starting with lower case, e.g. `balance` and `getBalance()`. Private fields should have a `.` (period) as a prefix to make it clear that it is a private field, e.g. `.balance`. Reserved keywords [10] and unsafe method names must also be avoided according to the RCC. The methods `setConstructorS3()` and `setMethodS3()`, described next, enforce these naming rules and if not followed, an *RccViolationException* is thrown. Not all rules are enforced in order to be backward compatible with some basic R functions that (for obvious reasons) do not comply with the RCC. As a last resort, it is always possible to turn off the test against RCC by using the argument `enforceRCC=FALSE` when calling the above functions.

4 Defining new classes

Under the S3 schema there is no way to formally define a class and there is no way to enforce that an instance of a class has the correct format or contains the correct fields, or to assure that the inheritance structure is valid. One reason for this is that the class of the object and the inheritance structure of the class is solely specified by the class attribute of the individual objects. This attribute may be modified in any way, at any time, which makes the object-oriented implementation vulnerable to programming mistakes, but also to misuse. The S4 schema over comes some of these lack-of-robustness drawbacks. The way the *Object* class is designed, the idea is that the fields (and hence the class) are defined inside the constructor function, and that the class attribute is never accessed by the programmer. This minimizes the risk for errors.

The two utility functions *setConstructorS3()* and *setMethodS3()*, introduced next, help the programmer to create constructors and methods without having to worry about generic functions. These functions can be used to define any method or classes, not only classes derived from *Object*.

4.1 Defining constructors

The *setConstructorS3()* sets the constructor function and automatically creates any necessary generic function (there are cases where this might be necessary). When defining a class descending from the *Object* class, the constructor function also plays the role of defining the class (its fields) and specifying which class to extend. For example, to create the SavingsAccount class we write:

```
setConstructorS3("SavingsAccount", function(balance=0) {
  if (balance < 0) {
    throw("Trying to create an account with negative balance: ",
                                                          balance)
  }
  extend(Object(), "SavingsAccount",
    .balance = balance
  )
})
```

The declaration of the inheritance is done via the *extend()* method of the *Object* class, which will be called recursively throughout all the superclasses. The first argument to *extend()* should be the object returned by the constructor of the superclass. In the above example, the SavingsAccount inherits directly from the *Object* class, which is done by calling its constructor. The second argument to *extend()* should be the name of the class to be defined, e.g. SavingsAccount. According to the RCC, the name of the class should be the same as the name of the constructor function. Any other arguments to *extend()* are optional, but they must be named value arguments, e.g. `.balance=balance`, which then declare the fields of the class and their default values. Finally, all classes derived from *Object must* comply with the rule that it should be possible to create an instance of it by calling its constructor with *no argument*[9], e.g. `account <- SavingsAccount()`, cf. *prototypes* in S4.

4.2 Defining methods

The *setMethodS3()* method creates methods for S3 classes and at the same time encapsulates a lot of details that the programmer should not have to think about. One such thing is if a generic function should be created or not, and if so, how it should be created. For a detailed discussion on how generic functions are automatically created if missing, see Section 4.4. To create the *setBalance(newBalance)* method for the SavingsAccount class, the only thing needed is:

```
setMethodS3("setBalance", "SavingsAccount",
                             function(this, newBalance) {
  if (newBalance < 0) {
    throw("Trying to create an account with negative balance: ",
                                                          balance)
  }
  this$.balance <- newBalance
})
```

[9] The reason for this is that *static class objects* are created by calling the constructor with no arguments.

The complete usage of *setMethodS3()* is:

```
setMethodS3(name, class="default", definition, private=FALSE, pro
tected=FALSE, static=FALSE, abstract=FALSE, trial=FALSE, deprecat
ed=FALSE, envir=parent.frame(), createGeneric=TRUE, enforceRCC=TR
UE)
```

where `name` is the name of the method, `class` is the name of the class, and `definition` is the definition, i.e. the function itself. If `class == "default"` (or `"ANY"`), a default function [10] is created, meaning that *setMethodsS3()* can be used whenever a function is defined. For all other arguments, see the help page of the function.

4.3 Details

The *setMethodS3()* method creates a standard S3 method and at the same time makes sure that a generic function for that method is available. For instance, the evaluation of

```
setMethodS3("getBalance", "SavingsAccount", function(this) {
  this$.balance
})
```

creates the S3 method for the class and the S3 generic function, i.e.

```
getBalance.SavingsAccount <- function(this) {
  this$.balance
}

getBalance <- function(...) UseMethod("getBalance")
```

It also makes sure that if there already exists a non-generic function called `getBalance()`, then it will be renamed to `getBalance.default()`. If the latter already exists, *setMethodS3()* cannot solve the conflict, and therefore an *Exception* will be thrown explaining this. A generic function is not created if a generic function (or an internal function that works as such) already exists. For instance,

```
setMethodS3("as.character", "SavingsAccount", function(this) {
  paste(data.class(this), ": balance is ", this$.balance, ".",
                                                          sep="")
})
```

creates only the S3 method and *not* the generic function. In addition to this, *setMethodS3()* will, by default, assert that the (most important) RCC naming rules are followed. If not, it throws an *RccViolationException* informing that an RCC rule was violated.

4.4 Safely creating generic functions

When writing a package it is important to make sure that the package does not overwrite preexisting functions. If a preexisting function exists that is not a generic function, in most cases, the conflict can be solved by redefining the function to become a default function. The test whether a function already exists or not is commonly done manually by the programmer. However, new functions might be added when a new version of R is released and more seriously, other packages might be loaded before or after our package is loaded and there is no way we can know which functions will be defined or not. A much safer approach is to check for conflicts and solve them when the package is loaded. Furthermore, it is important to make sure that the generic function will work with all packages and not just the methods in our package. Moreover, complete object-oriented programming requires that methods can have the same name for different classes, but with another set of arguments. By not specifying the arguments of the generic functions, but only the special ... argument, the generic function is "as generic as possible", e.g. `getArea <- function(...)  UseMethod("getArea")`[10]. For a further discussion on how to create generic functions safely, see [4]. These problems are all taken care of automatically by *setMethodS3()*.

5 The root class Object

By enforcing that all classes are derived (directly or indirectly) from a common root class, we know that there exists a set of methods that are applicable to all such classes. This idea already exists to some extend in R, but by using a common root

[10] If we *do* specify any arguments, we restrict the corresponding methods for all classes in all packages loaded at the same time to have the exactly the same set of arguments. Under the S4 style, it is required and enforced that *all* methods for all classes have exactly the same arguments as the corresponding generic function. This is one of the reasons why we currently not use the S4 style of programming with classes, but we hope to overcome this problem soon by making use of namespaces.

class it will be more explicit to the end user. The R.oo package defines the root class *Object*, which has the following methods coupled to it. See also Figure 2.

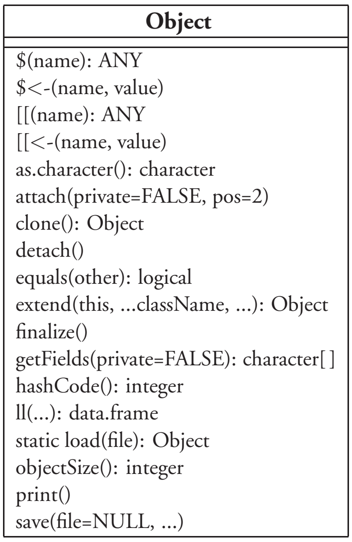


Figure 2: UML representation of the root class *Object*, which all classes should be derived from directly or indirectly through other classes. ANY is not a defined data type, but refers to any data type or class.

The method ***as.character()*** returns a string with short information about the Object. This is the same string that by default is displayed by *print()*.

The ***print()*** method prints information about the Object. By default, the string returned by *as.character()* is printed. For convenience in R, *any* object of any data type or class whose name is typed at the command line followed by ENTER, the *print()* of that object will be called. Example:

```
> 1+2      # gives the object '3', i.e. print(3)
[1] 3
> account  # same as print(account)
SavingsAccount: balance is 100.
```

The method ***getFields(private=FALSE)*** returns the name of all fields in the Object. By default, only names of non-private fields are returned.

The method ***ll(...)*** returns a data frame with detailed information about the fields of the Object. By default, only non-private fields are listed. For more information see section 8.

The ***hashCode()*** method returns an integer hash code for the Object.

The ***objectSize()*** method returns the (approximate) size of the Object.

The ***equals(other)*** method compares one Object with another. If they are equal, the method returns `TRUE`, otherwise `FALSE`. If argument `other` is `NULL`, then `FALSE` is always returned. The default implementation of *equals()* is comparing the *hashCode()* values of both objects.

The ***clone()*** method creates an identical *copy* of the Object[11].

When an Object is deallocated from memory by the garbage collector the ***finalize()*** method is first called. Subclasses can override this method to make sure that any instances of those classes clean up after themselves. For instance, objects that allocate shared resources such as connections should make sure that these resources are closed and deallocated upon deletion.

The methods ***attach(private=FALSE, pos=2)*** and ***detach()*** attach and detach an Object to the search path, respectively. By default, only public fields (private=FALSE) of an Object are attached, and, by default, they are attached to the beginning of the search path (pos=2) immediately after the global environment. Any modification to such attached fields will *not* be reflected (saved) in the actual Object. The attach-detach mechanism is intended for read-only purposes.

The method ***save(file=NULL, ...)*** saves an Object to a file (or a connection), and the *static* method ***load(file)*** loads a previously saved Object and returns a reference to it.

The somewhat special method ***extend(...className, ...)*** *extends* an *Object* class into a subclass named according to the string ...className[12] and which contains

[11] Doing `ref2 <- ref` will only create a new reference to the same instance.

[12] The second argument to extend() has three dots as a prefix to make it possible to name fields

all fields as given by the ... arguments. This method is not intended to be overridden by any subclass.

Finally, as explained in the next section, the functionality for references is hidden inside the *Object* class. Hence, all subclasses will support references automatically and the programmer does not have to think about how reference variables should be implemented. They are always provided and they always behave in the same way.

6 Reference variables

All instances of the *Object* class (or one of its subclasses) are accessed via *reference variables* or shortly *references*[13]. In standard R, where reference variables are not provided, each instance of a class is accessed by one single variable, the object itself. With references, however, it is possible for several variables to access the same object. Here is an example where a list contains several references to the same Object:

```
person <- Person("Dalai Lama", 68)
l <- list(a=person, b=person, c=clone(person))
setAge(l$a, 67)
print(person)
[1] "Dalai Lama is 67 years old."
setAge(l$c, 69)
print(person)
[1] "Dalai Lama is 67 years old."
```

If `person` would *not* be a reference, the two elements `a` and `b` would be another two copies (clones) of the Person object and a modification of one of them would not affect the other instance and neither the original variable `person`. It is possible to create a copy of an Object by using `clone()` as illustrated in the above code.

as className or similar.

[13] For those who are not familiar with references but with pointers, references can be thought of as safe pointers to objects that cannot by mistake be made to point to the wrong part of the memory. Object-oriented programming where objects are passed by references does not differ much from the case when objects are passed by value. However, there are differences that are important to be aware of.

Furthermore, references make it possible to implement software that otherwise would not be possible, or would be very tedious, to implement. Using references, more details can be encapsulated and thereby the package will be more user-friendly. We believe that a well designed object-oriented method interface based on references can serve as a base for, but also be a good complement to, a graphical user interface (GUI). For more complete real-world examples, see [2] and [1].

6.1 Garbage collector

The use of references requires a memory management. Many languages, including R, provide a built-in *garbage collector*, which removes obsolete objects from the memory that are not referred to by anyone. Since objects inherited from the *Object* class are standard R objects they are recognized by the garbage collector. For example, an Object created inside a function and for which no reference is returned, will be deleted by the R garbage collector. In summary, objects do not have to be deleted explicitly, but for an Object to be deleted it is important that all references to it are removed, for instance by *rm()*, or set to `NULL`. It is a good custom to do this as soon as an Object is not needed.

6.2 Details

R does *not* support references, but references can be emulated using so called *environments* [10]. However, using environments explicitly will quickly fill the source code with several `get(name, envir=ref)`, `assign(name, value, envir=ref)` and/or `eval(..., envir=ref)` statements. This makes the code hard to read and increases the risk for errors. By *encapsulating* all calls to *get()* and *assign()* in the operator methods *$()*, *[[()*, *$<-()* and *[[<-()* of the root class *Object*, all fields can be accessed as if they were elements in a list. There are other ways for emulating references and some are more and some are less memory and time efficient than others. The R.oo package use the first approach, where each object lives in its own environment. One reason is that the garbage collector recognizes environment variables.

7 Exception handling

In addition to methods for defining classes and support for references, the package provides an improved *exception handling* mechanism. The core functionalities for exception handling is done by the *Exception* class (see Figure 3). It provides methods to create and throw exceptions and together with its companion *trycatch()* complete exception handling is provided.

7.1 Creating and throwing exceptions

The easiest way to create and throw an exception is by calling *throw()*, e.g.

```
throw("Division by zero.")
```

which is equivalent to calling

```
throw(Exception("Division by zero."))
```

An object of any class that inherits from *Exception* contains information about the error and when it occurred. Any *Exception* object can be thrown using the *throw()* method and then optionally be caught by either *trycatch()* or *try()*. If an *Exception* is thrown, the last exception thrown can be obtained by the static method *getLastException()* of class *Exception*. The *as.character()* method for the *Object* class is overridden by the *Exception* class and the default print message of an *Exception* has the format:

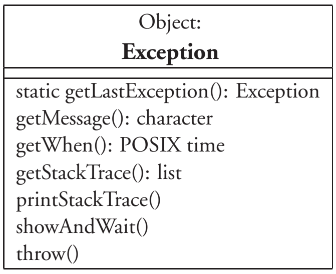


Figure 3: UML representation of the *Exception* class, which extends the *Object* class.

```
> throw("Division by zero.")
Error: [2002-10-20 10:24:07] Exception: Division by zero.
```

7.2 Catching exceptions depending on class

The *trycatch()* method can catch exceptions based on what class they belong to. Like the *try()* function, the first argument to *trycatch()* is the expression to be evaluated, which might throw an exception. Any further arguments must be named arguments where the name specifies the *Exception* class to be caught and the value the code to be evaluated if such an exception is thrown. An argument with name `ANY` will catch any kind of Exception (including `try-error` thrown by `stop()`). If an exception is caught, and no further exceptions are thrown, then `trycatch()` returns safely. The following code will generate and throw an exception, which will be caught by the `ANY` clause, preventing the R session from being interrupted.

```
trycatch({
  x <- log(2)
  y <- log("a")
}, ANY={
  x <- 0
  y <- 0
  print(Exception$getLastException())
})

print("trycatch() did indeed catch the exception.")
```

Moreover, code defined by an argument named `finally` is guaranteed to be evaluated immediately before *trycatch()* returns. This is, for instance, useful if a connection needs to be closed regardless of whether an exception is thrown or not.

8 Utility functions

In addition to the already mentioned methods, the package also defines some useful utility functions, which are applicable to objects of any class or data type. The default method of ***ll()*** lists detailed information about objects (variables and functions) found in an environment. The returned data frame will by default contain information about the *member* (name of the variable or function), *data.class*, *dimension* and *object.size*, which are the values returned by the functions with the same names.

```
> ll()
       member   data.class dimension object.size
1     analyze     function      NULL         248
2          ma       MAData         1         452
3         raw      RawData         1         452
4         gpr  GenePixData         1         460
5           y      numeric       100         828
```

For information about other utility functions provided by the package, see the help of the package.

9 Other classes

Other classes that are loaded with this package are *Class*, *Package* and *Rdoc*. The class *Class* provides an interface for querying classes about methods, fields etc. The class *Package* represents any kind of package, e.g. `Package("base")`. Given a *Package* object, it is possible to query it for its classes, its author, check for updates (see below for an example) etc. The *Rdoc* class provides a compiler for *Rdoc documentation*, which is an extension of the Rd language that minimizes the need for having to update the documentation when the source code is updated. For instance, the tag `@synopsis` generates a correct `\usage` (or a `\synopsis`) markup given the other information in the Rdoc code. The Rdoc documentation can be standalone files similar to Rd files (the simplest Rdoc file is a plain Rd file) or it can be part of the source files in form of comments. The Rdoc code is compiled into standard Rd files, which are then converted into help pages etc. by `R CMD build`. We intend to extend the Rdoc compiler to recognize S4 classes too.

10 Installation

Since the package was written in 100% R, no native code needs to be built and installation is straightforward. The R.oo package is part of a bundle of packages called R.classes [5]. To download and install the *R.classes* bundle, do

```
install.packages("R.classes", contriburl="http://www.maths.lth.se
/help/R")
```

from within R. By default, the R library directory is the directory named `library/` in the directory where R is installed. The bundle can be installed in a private directory by setting the environment variable `R_LIBS`, e.g. `setenv R_LIBS $HOME/R/`. By loading the package, e.g. `library(R.oo)`, the correctness of the installation can be verified. To install on a Macintosh that does not have OS X, or to manually install the bundle, see [5]. For future updates, load the package and do

```
update(R.oo)
```

11 Conclusions

The R.oo package is open source, it is designed in an object-oriented style and implemented using plain and richly commented R code (100% R). Moreover, it is designed and implemented such that any future migration from S3 to S4 will be as smooth as possible for the end user.

For over two years we have used the R.oo package and the *R.classes* bundle in a project developing a cDNA microarray analysis package (com.braju.sma [1]). We have found that, by using the R.oo package, we never have had problems with conflicts related to generic functions and we never have had to create a generic function explicitly. In cases when we by mistake tried to use a reserved word as method name, *setMethodS3()* immediately notified us. We have also found the Rdoc compiler to be a valuable tool for maintaining nearly 200 Rd files. Due to the huge memory load and the large amount of redundancy in microarray data, the use of reference variables has been a natural and successful choice. Moreover, since methods can change the state of objects if references are used, we have been able to decrease the number of arguments that has to be specified in the method calls and therefore we can provide a cleaner and more user-friendly method interface. For a further discussion on how the R.oo package has been used in the development of our microarray package, see [1]. To install the com.braju.sma package see [1].

References

[1] Henrik Bengtsson. com.braju.sma - object-oriented microarray analysis in 100% R. http://www.maths.lth.se/help/R/, 2002.

[2] Henrik Bengtsson. The com.braju.sma package - a microarray analysis package based on an object-oriented design and reference variables, 2002.

[3] Henrik Bengtsson. Programming with references - a case study using the R.oo package, 2002.

[4] Henrik Bengtsson. Safely creating S3 generic functions using setGenericS3(). http://www.maths.lth.se/help/R/, 2002.

[5] Henrik Bengtsson. The R.classes bundle (R.oo and friends), 2003.

[6] John M. Chambers. *Programming with Data*. Springer, 1998.

[7] John M. Chambers. S language methods and classes, 2002.

[8] John M. Chambers and Duncan Temple Lang. OOP programming in the S language, 2002.

[9] Object Management Group. UML Resource Page, 2002.

[10] R Development Core Team. R Language Definition (v1.7.0, draft), April 2003.

[11] R Development Core Team. Writing R Extensions (v1.7.0), April 2003.

[12] W.N. Venables and B.D. Ripley. *Modern Applied Statistics with S-PLUS*. Springer, third edition, 1999.

D

Paper D

Methodological study of affine transformations of gene expression data with proposed normalization method

Henrik Bengtsson[1] and Ola Hössjer[2]

[1] Mathematical Statistics at the Centre for Mathematical Sciences, Lund University, Box 118, SE-221 00 Lund, Sweden. Email: hb@maths.lth.se.
[2] Mathematical Statistics at the Mathematics Department, Stockholm University.

ABSTRACT

A detailed methodological study of affine models for gene expression data is carried out. We focus on two-channel comparative studies, although the findings are not restricted to such. We find that an affine model is capable of explaining many of the commonly observed discrepancies and systematic effects in gene-expression levels and observed log-ratios. Focus is also on data obtained by the two-color spotted microarray technology, but most of the discussion applies equally well to other gene-expression techniques such as single-channel hybridization methods and quantitative real-time PCR. An affine model can also explain non-linear systematic effects commonly observed when log-ratios obtained by different gene-expression technologies are compared. A high-quality cDNA microarray data set is used to demonstrate the power of the affine model. In the light of the affine model, the strengths and the weaknesses of the most commonly used normalization methods are discussed. Based on the affine model, we propose a novel method to normalize one or multiple arrays simultaneously, where each array has been hybridized with one, two or more samples. A package with all necessary methods to read and normalize data have been written in the R language and is made available for free.

Keywords: microarrays; affine transformation; bias; logarithm; non-linear systematic effects; robust normalization; background correction; principal component analysis; iterative reweighted PCA.

1 Introduction

The general idea behind most gene-expression measurement technologies is to assess the *expression levels* of (all or a subset of) genes in one or several cell populations. The ensemble of gene-expression levels is often referred to as the *gene-expression profile* of the sample. As it is not possible to measure the expression level of a gene in practice, techniques have been developed to measure closely related properties such as the amount of mRNA (messenger RNA) or the amount of proteins produced. Although different techniques to measure protein levels have evolved and improved recently, they are still not as cheap and widely used as those to measure mRNA levels. The amount of mRNA is either measured directly or, as it is much less stable than DNA, indirectly by first translating it into cDNA (complementary DNA). One of the technologies measuring cDNA is Southern blot, which is based on electrophoresis in gels and measures the DNA levels of a small set of genes simultaneously. A succeeding technology, Northern blot, is similar, but measures the RNA levels instead. Another technology that measures mRNA levels indirectly via cDNA is QRT-PCR (quantitative real-time polymerase chain reaction). QRT-PCR is a low-throughput technique, but, because of its high accuracy (closer to the truth) and high precision (more similarity between replicated measurements), it is often taken as the *gold standard*. For this reason, QRT-PCR is often used to verify the correctness of other technologies such as the microarray technology.

In addition to the above methods, recently developed *microarray techniques* [30] provide a way of measuring the expression levels of many more genes, in the order of $10^3 - 10^5$ or more. This has made it possible to take a snapshot of the activity of a person's complete genome in one single experiment. Microarrays have well defined and immobilized regions that each consists of clones or synthesized sequences of DNA specific to a unique gene. We refer to these (non-hybridized) regions or spots as *probes* [13]. A cocktail of cDNA created from the RNA extract, referred to as the *target*, from the cell population in study is then *hybridized* to the DNA on the microarray for a few hours after which excess cDNA is washed off. The result is that each region of the microarray contains a certain amount of hybridized DNA unique to the corresponding gene. By first labeling the cDNA strands in the sample cocktail with a radioactive or a fluorescent probe, the amount of hybridized DNA can be measured using radioactive sensitive film or a color-sensitive scanner, respectively.

When measuring gene expressions for an individual gene we try to assess how active (measured on some scale) that gene is. Because it is hard to identify an *absolute* scale to measure on, often, but also for various other reasons, a reference is used to obtain a *relative* scale. As even genes from the same sample are not directly comparable to each other, each gene gets its own reference, which is typically the same gene from a reference sample. With this approach, we can obtain *gene-expression ratios* for every gene, which for instance can be used to test the hypothesis if a gene (in the test sample) is *differentially expressed* or not (compared to the gene in the reference sample). This is the core idea behind the two-channel microarray technology, in which the test and the reference cDNA cocktails are hybridized to the same array simultaneously and in a competitive way. The same idea has been adopted by single-channel hybridization technologies where the comparison is instead done numerically in the data analysis step.

Even if gene-by-gene references are used, the measurements are not perfect and are likely to contain systematic errors, which possibly vary from measurement to measurement, and the obtained gene-expression ratios may still be biased and not comparable to each other. What we ultimately would like to do is to measure all control and all reference samples under identical conditions. The aforementioned two-color microarray technology tries, in some sense, to do this by measuring the control/reference pairs for each gene in one hybridization (although it is not clear if the gain from co-hybridizing two samples with different labels is larger than hybridizing twice with identical label molecules and then scanning the samples separately).

In this paper we present an affine model, which can explain many of the systematic effects often observed when gene-expression levels from two (or more) samples are compared. The main contributors to such systematic effects are the biases in each channel, which give non-linear systematic effects in ratios etc. Here we will not give an error model, but instead only a deterministic model. The main reason for this is that we believe an error-free model makes us better understand the impact that channel biases have on the downstream analysis regardless of gene-expression technology used. This is especially of interest as these are often implicitly assumed to be small and of no effect, which we believe is a too strong assumption. Although we do focus on the microarray technology and although some results have been published [28], we believe much more research is needed before we can understand and correctly model the various error sources introduced in the microarray process.

The outline of this report is as follows. In Section 2, a general model that incorporates all steps of any gene-expression technology is given. By dissecting the generic model and focusing more on the microarray technologies, an affine model is introduced. In Section 3, we study the properties of the widely adopted and accepted log-ratio log-intensity transform under the affine model. An investigation of how the affine transform introduces biases in the log-ratios is carried out. In Section 4, we revisit some of the normalization methods available in literature, to which background correction may also be counted. We conclude the section by suggesting a novel robust normalization method that can be applied to single-channel as well as multi-channel microarray data. The method will correct for both systematic effects and scale differences in the log-ratios. We end the report with a discussion on similarities with other normalization method, which gives us some future research directions.

2 Model

2.1 A general model

Consider an experiment that involves genes $i = 1, 2, \ldots, I$ from RNA extracts $c = 1, 2, \ldots, C$. For example, in oligonucleotide microarrays each slide measures the gene-expression levels of exactly one RNA extract whereas for a two-color microarrays each slide measures two RNA extracts, one in each channel. From now on, we refer to the RNA extracts or replicates of such as *channels*. Let $x_{c,i}$ be the true gene-expression level of gene i in channel c and let $y_{c,i}$ be the corresponding observed gene-expression level. The relation between the observed and the true expression levels can be written as

$$y_{c,i} = f_c(x_{c,i}) + \varepsilon_{c,i} \tag{1}$$

where $f_c(\cdot)$ is a channel specific function, which we refer to as the *measurement function*, that includes all steps in the gene-expression acquisition process. Most generally, we have that $E[\varepsilon_{c,i}] = 0$ and $V[\varepsilon_{c,i}] = \sigma^2_{c,i}$, where the variance can take any form. Importantly, the properties of ε_c are still not well known and varies depending on if the overall or a submeasurement function is considered. For instance, under certain circumstances it can be shown that the standard deviation of the noise is proportional to the signal, but in other cases the relationship might be more or less complicated. For this reason and because of the many interesting effects that the affine transformation (presented below) generates by itself, we have decided to conduct this study under the assumption of noise-free data.

To continue, since we want to do inference based on $x_{c,i}$, the inverse of $f_c(\cdot)$ has to be identified, something that is possible if it is strictly increasing. Violation of this constraint has been observed in, for instance, two-color microarray data. This can be due to too high concentrations of fluorophore molecules, which quenches the signal, sometimes so much that the signal decreases when the concentration increases [27, 24]. Extreme saturation in the scanner, which is commonly observed when the PMT gain is set too high, results in censored signals, which in turn prevents us to find a unique inverse of the measurement function. The above relationship can be specific for certain subsets of genes or spots such as print tip [35], microtiter plate or clone library [4], spatially dependent or even gene specific, but in the general discussion that follows we will avoid such details for simplicity.

2.2 Dissection of the overall measurement function

Gene-expression levels are not measurable directly and any technique used transforms the expression level into a measurably quantity, commonly a fluorescent intensity or a voltage level. Mathematically, each step in the microarray process can be seen as a function that takes a set of input objects and outputs another set of objects. The sequential nature of the process allows us to think of the measurement function $f_c(\cdot)$ as a *composite function* (function of functions); $f_c(\cdot) = (f_{c,K} \circ f_{c,K-1} \circ \cdots \circ f_{c,1})(\cdot)$, where K is the number of steps in the process. For instance, and of course simplified, it could be that $f_{c,1}(\cdot)$ models the extraction of the RNA from the cell, $f_{c,2}(\cdot)$ models the reverse transcription of RNA into cDNA etc., and $f_{c,K}(\cdot)$ models the scanning and the signal quantification of the hybridized array. We realize that some of these *submeasurement functions* are shared by several channels and that others are channel specific or even gene specific. Moreover, there is also one or several joining subfunctions, e.g. the hybridization of labeled cDNA sequences to the probes on the array. Thus, the overall measurement function $f_c(\cdot)$ consists of a complex and to a large extent unknown ensemble of subfunctions $\{f_{c,k}(\cdot)\}_{k=1}^{K}$.

A first order Taylor series expansion of an arbitrary measurement function $f_c(x_c)$ has the form

$$f_c(x_c) = a_c + b_c x_c + R_c(x_c),\ \forall c. \tag{2}$$

Next, we will argue that a_c cannot be assumed to be zero and b_c cannot be assumed to be one. In addition, in order to focus on the effect of a_c and b_c, we will ignore higher order terms ($R_c(x_c)$) assuming they are small. Thus, we do not take into account the possibility that saturation might occur. However, as we believe saturation can be avoided, we will not discuss function of higher order in this report, but instead we focus on measurement functions that are affine.

2.3 Existence of biases in each of the channels

From the above dissection of the measurement functions it is not hard to believe that some of the subfunctions introduce bias and therefore we ought to have an overall bias in $f_c(\cdot)$. For instance, the bias terms can be due to non-uniformity of the reverse transcription, the labeling [24] or the hybridization, due to dark noise in the PMT [19] or laser scatter light in the scanner, background noise, or non-uniformity of the scanned glass slide [16], threshold effects etc. In [3] it is shown how various background estimates based on different image analysis methods may introduce bias. Similarly, we have observed that different scanners, but also different image analysis methods, introduce a bias in the observed signals [6].

2.4 Existence of scale factors in each of the channels

It is not hard to imagine that there is also an overall scale factor (different from one) in each $f_c(\cdot)$. This can be due to differences in labeling or hybridization efficiency, laser power, PMT gain, wavelength dependent optical effects, bleaching of fluorophores, and so on. As in the case for the biases, the scale may vary between different subsets of genes or spots as well as with spatial position. Different scale factors have also been observed between clone libraries and spike-in controls [32].

2.5 The affine measurement function

From the above discussion the simplest possible parametric measurement function reasonable to consider is an affine transformation of the RNA concentration $x_{c,i}$ as

$$f_c(x_{c,i}) = a_c + b_c x_{c,i},\ \forall c, i, \tag{3}$$

with unique inverse

$$x_{c,i} = f_c^{-1}(y_{c,i}) = \frac{y_{c,i} - a_c}{b_c},\ \forall c, i, \tag{4}$$

where a_c is the overall bias and b_c is the overall scale factor of channel c. Both the bias and scale parameters are commonly positive, but under certain circumstances, for instance, as demonstrated later, when two different measuring techniques are compared, the effective bias may be negative. Recall that especially a_c, but also b_c, may vary between different subset of genes etc., but for simplicity we focus on the case where all spots have the same bias. The left plot in Figure 2 shows, on a log-log scale, an example of an affine transformation of two-color microarray data where the offsets in the red and the green channels are 20 and 200, and the scale factors are 0.8 and 1.4, respectively. Modeling microarray data by an affine transform is not novel [28, 17, 21, 12], but the reasons for such a model might have been different in those papers.

3 The log-ratio log-intensity transform

Especially in two-color but also in oligonucleotide microarray experiments, it is convenient to do statistical analysis on the log-ratios and the log-intensities [8] of the gene-expression levels in two channels instead of on the expression levels directly. For gene i we have that

$$M_i = \log_2 \frac{y_{R,i}}{y_{G,i}} = \log_2 \frac{f_R(x_{R,i})}{f_G(x_{G,i})} \tag{5}$$

$$A_i = \frac{1}{2}\log_2(y_{R,i} \cdot y_{G,i}) = \frac{1}{2}\log_2(f_R(x_{R,i}) \cdot f_G(x_{G,i})). \tag{6}$$

For clarification (and for simplicity) we denoted the two channels by R and G instead of 1 and 2, which are mnemonics for the red and the green dyes commonly

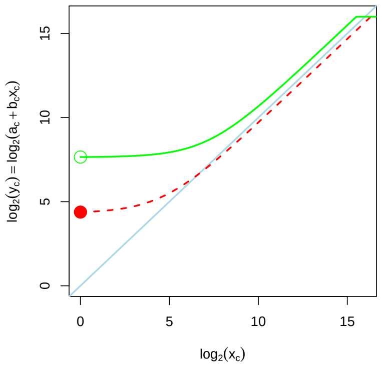


Figure 1: Log-log plot of the transformed red (dashed) and green (solid) signals as a function of the true signals under the affine transformation $(a_G, a_R) = (200, 20)$ and $(b_G, b_R) = (1.4, 0.8)$. The light blue solid line represents the $x_R = x_G$ line. Since $b_G > 1$, $f_G(x_G)$ will be saturated at high intensities.

used in two-color microarray data. It may also be of interest to compare log-ratios within the same channel, which is the case for instance in single-channel normalization and calibration methods [7, 6, 37, 32]. One of the rationales for this bijective transform (if the observed signals are positive) is that the main measure of interest, the fold change, is contained in one variable. However, since the transform is based on observed expression levels and not the true ones, the conclusion that all information about the biological fold change is contained in M only is not necessarily true. This can be seen if we consider the true fold change for an arbitrary gene i

$$r_i = x_{R,i}/x_{G,i} \tag{7}$$

where $r_i > 0$. If we drop the gene index i in (5) and (6), M and A can be written as functions of x_G and r, i.e. $M = g_r(x_G)$ and $A = h_r(x_G)$. Now, we can write

$$M = m_r(A) = g_r(h_r^{-1}(A)). \tag{8}$$

It is not hard to see that the log-ratios M are independent of the log-intensities A for all fold changes if and only if $f_R(\cdot) = f_G(\cdot)$. Hence, and discussed thoroughly below, commonly observed intensity-dependent effects in the log-ratios may contain valuable information, and consequently, applying normalization methods without care may result in loss of information and introduced bias.

3.1 Log-ratios as a function of log-intensities

Under an affine transformation one can show (appendix A.2) that the relationship between the observed log-ratios and the observed log-intensities for fold change r is

$$M = m_r(A) = \log_2 r + M_{\text{bias}} \tag{9}$$

$$M_{\text{bias}} = \log_2 b + \log_2 \left(\frac{\frac{1}{2}(a_R - rba_G) + \sqrt{\frac{1}{4}(a_R - rba_G)^2 + rb2^{2A}}}{-\frac{1}{2}(a_R - rba_G) + \sqrt{\frac{1}{4}(a_R - rba_G)^2 + rb2^{2A}}} \right) \tag{10}$$

where $b = b_R/b_G$ is the *relative scale factor* between the two channels compared. Recall that $log_2\, r$ (the true log-ratio) is the variable of interest. The derivative of M with respect to A for a fixed fold change r is

$$\left.\frac{dM}{dA}\right|_{x_R = rx_G}(A) = -\frac{a_R - rba_G}{\sqrt{\frac{1}{4}(a_R - rba_G)^2 + rb2^{2A}}}. \tag{11}$$

Regarding r as fixed, there are only two parameters in (9)-(11) that determine the shape of the function $m_r(A)$. The first is $b = b_R/b_G$ and the second is $a_R - rba_G$. Consequently, when $a_R, a_G \neq 0$, M is independent of A if and only if $r = (b_G a_R)/(b_R a_G)$. For this particular value of r, we have that the observed log-ratio is $M = \log_2(a_R/a_G)$, which is independent of the scale factors. Moreover, for log-ratios of non-differential expressions, that is $log_2\, r = 0$, to be independent of A, it must be true that $b_G a_R = b_R a_G$ or, equivalently, $b_R/b_G = a_R/a_G$. It is also clear from Equation (9) and Equation (11) that the scale parameters cannot introduce any curvature themselves, which was incorrectly argued in [12], but only enhance or suppress curvature introduced by the bias. In addition to this, the relative scale will shift the log-ratios up or down. In the case of a linear transform ($a_R = a_G = 0$) we notice that M is independent of A for any choice of r and that the bias in the log-ratios is $\log_2(b_G/b_R)$. Moreover, the size of the effect that the bias terms have on the log-ratios decreases as the intensity increases. At high intensities the only observable effect is that from the relative scale between the two channels. The observed log-ratio for non-differentially expressed genes at high intensity is $M_\infty \approx \log_2 b$. The weakest data point possible to observe is $(A_0, M_0) = (1/2 \cdot \log_2(a_R a_G), \log_2(a_R/a_G))$, which is independent of both gene expression and scale parameters. If $a_R, a_G > 0$, all fold-change curves will converge to this point. In the left graph in Figure 2 the

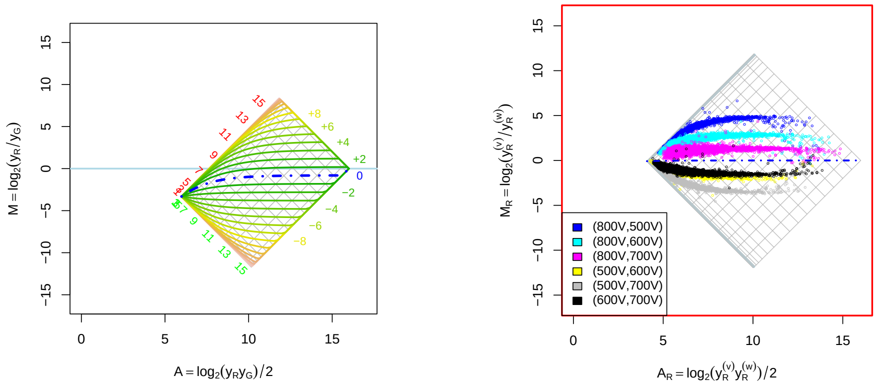


Figure 2: Affine transformation of the red and the green signals where $(a_G, a_R) = (200, 20)$ and $(b_G, b_R) = (1.4, 0.8)$. *Left:* The observed log-ratios as a function of the observed log-intensities for different fold changes. The blue dot-dash curve corresponds to the non-differentially expressed genes and the thinner curves above and below this curve represent $\log_2 r = \pm 1, \pm 2, \ldots$ as labeled to the right of the curves. The lines in the gray grid, which is rotated 45 degrees, show the levels where the *true* signals $\log_2 x_R$ and $\log_2 x_G$ are equal to $\ldots, -1, 0, 1, \ldots, 16$. These levels have been labeled to the left of the grid. No observations can lie outside this grid. *Right:* Real-world example of an affine transformation. The same slide was scanned four times at four different PMT settings. For each of the six scan pairs, the *within-channel* log-ratio and log-intensities were calculated. Data shown is from the red channel, which was estimated to have an offset of $a_R = 20.3$ for all scans.

effect of the affine transform $(a_G, a_R) = (200, 20)$ and $(b_G, b_R) = (1.4, 0.8)$ at different fold changes is depicted. The different curves plotted are the functions $M = m_r(A)$ for different fold changes. Non-differentially expressed genes

($\log_2 r = 0$) lie along the blue dot-dash curve, two times up and down regulated genes ($\log_2 r = \pm 1$) are the thinner curves above and below the non-differentially expressed curve. The curves outside these represent $\log_2 r = \pm 2, \pm 3, \ldots$. Note the asymmetry in curvature between up and down regulation. From the above discussion we know that the observed log-ratios are independent of the log-intensities for $log_2 r \approx -2.51$ with the value $M_0 \approx -3.32$. The log-ratios for non-differentially expressed genes at high intensities is $M_\infty \approx -0.81$. A real-world example taken from [6], where the same array was scanned four times at various scanner PMT (sensitivity) settings, is shown in the right plot in Figure 2. Observed within-channel log-ratios $M_c = \log_2(y_c^{(v)}/y_c^{(w)})$ are plotted against the within-channel log-intensities $A_c = \log_2(y_c^{(v)} y_c^{(w)})/2$ for the red ($c = R$) channel where $y_c^{(v)}$ and $y_c^{(w)}$ are observations at two different scanner PMT settings. In this case it turned out that all scans share the same bias. For more details see [6].

3.2 Bias in the log-ratios

From Equations (9)-(10), we see that the bias in the log-ratios introduced by the affine transform is intensity dependent. This non-linearity can be observed as a propeller shaped graph in Figure 3, where the log-ratios under the affine transform $(a_G, a_R) = (200, 20)$ and $(b_G, b_R) = (1.4, 0.8)$ are plotted against the true log-ratios at different log-intensity levels. When doing a linear regression between the affinely transformed log-ratios and the true log-ratios, the slope will always be *less* than one. Moreover, this will be true for all normalization methods that do not overcompensate for bias. This can explain why some studies show that cDNA microarrays tend to compress the absolute log-ratios compared to oligoarrays and QRT-PCR [38, 39, 2]; the channel biases in cDNA microarrays are probably larger. When [23] compared cDNA microarray log-ratios with Northern blot log-ratios for their background correction method they found similar behavior, which emphasizes the close relationship between bias and background estimates. We will return to this later. Furthermore, note that the relationship between observed and true log-ratios does not become one to one for large fold changes. The reason for this is partly explained by the fact when the fold change increases the signal in one channel increases, but it could also be that the signal in the other channel decreases. In such cases, the latter signal will be affected relatively more by the bias, which introduces a non-linear relationship. This effect can be observed at the extreme fold changes in the left graph in Figure 3. The same patterns can be seen if we do an M versus M scatter plot for non-normalized versus (affinely) normalized data. See right scatter plot. To visualize the intensity dependency of the log-ratios, only data points at certain log-intensity levels are plotted. For details on data, see [6].

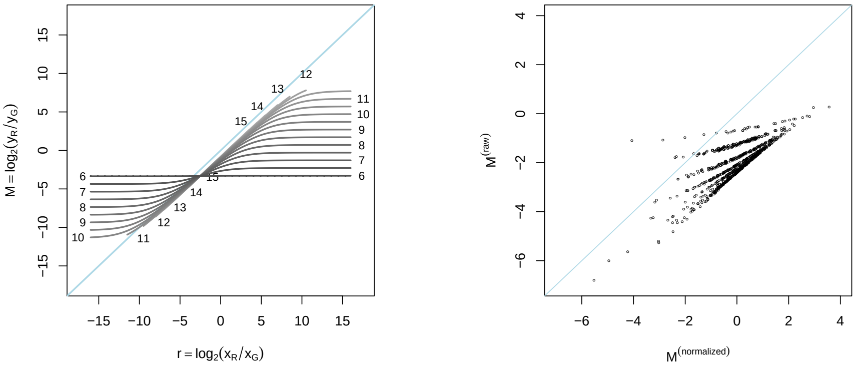


Figure 3: *Left:* Bias in the log-ratios introduced by the affine transform $(a_G, a_R) = (200, 20)$ and $(b_G, b_R) = (1.4, 0.8)$. Each line displays the relationship between the observed and the true log-ratios at a certain (observed) log-intensity A. Each curve is marked with the value of A. We have chosen to truncate the curves when the signals become saturated and the labels for those curves are positioned approximately where they have been truncated. For low intensities there is a great bias (deviance from the diagonal line), especially for large fold changes. At higher intensities the bias is smaller. The curves intersect at the one fold-change level that is independent of the intensity. *Right:* Real-world example of log-ratios for non-normalized versus (affinely) normalized signals. To clarify the intensity-dependent effect only data points close to $A = 5.0, 5.5, \ldots, 16$ are shown.

4 Normalization

Depending on the design of the microarray experiment different types of patterns in data are expected to be observed. A typical example is where a subset of the genes studied is expected to be non-differentially expressed in a test sample compared to a reference sample. It is common that the meaning of *subset* is "most genes", "house-keeping genes", or similar. Furthermore, some genes are expected to not be expressed at all, in one or both channels. However, it is common that the patterns of the *observed* expression levels are not in line with the expected patterns of the *true* expression levels. Whenever this happens different approaches can be taken to make the observed data meet the expectations. Normalization of microarray data is about identifying and removing such artifactual variations that are not due to noise or natural variability. An example is the intensity-dependent log-ratio artifact, which we already have shown can, to a large extent, be explained by biases in the different channels.

In this section we will, with the affine model in mind, revisit various more or less well known normalization methods that directly or indirectly remove intensity-dependent artifacts. The most well known is the lowess or curve-fit normalization method that, based on the assumption that most genes are non-differentially expressed, fits a curve through data and shifts the signals such that the log-ratios lye along the $M = 0$ line. Other models that rely on the same assumption, but normalize data differently, are the perpendicular and parallel translation normalization methods. These are discussed below. Although they are design oriented, we will also discuss various so called dye-swap normalization methods. More sophisticated are the quantile normalization methods, which rely on the assumption that the distribution of the gene-expression levels should be the same for all channels and arrays. These can also be understood in the light of the affine measurement function. With the gained knowledge we will then propose a generic and robust normalization method based on the affine model.

To be more precise in what follows, we will refer to methods that correct for differences in observed and expected data, that is, conform the signals to a standard or a norm, as *normalization methods*, where normalization has the meaning of conforming to expectations. Sometimes *calibration data*, also known as control data, which contains true relative or absolute expression levels, is available. Such data can be used to correct for discrepancies between the observed and the true expression levels. We refer to methods that use calibration (read *known*) data points to correct for artifacts as *calibration methods*. To this category we also count methods that are based on models for which we can find the inverse of the measurement function. We will not discuss calibration methods here.

In general, a normalization method is only capable of estimating $a_R - ba_G$ in Equation (10) and not the individual bias terms. This is because the *assumption that most genes are non-differentially expressed* (and/or that there is an equal amount of up and down regulated genes) will only help us identify one fold-change curve, namely $\log_2 r = 0$. For a normalization method, like most calibration methods, to be able to estimate both a_R and a_G more constraints are needed and without known data this can only be done based on more assumptions. As more research is needed, we will not elaborate with such additional assumptions in this report. Thus, the rest of this report will only discuss normalization methods based on the commonly accepted assumption that it is possible to identify a set of genes that can be used to normalize the non-differentially expressed genes.

4.1 Curve-fit normalization revisited

When [35] first observed the intensity-dependent effects on the log-ratios they suggested a curve-fit normalization method that is often referred to as *lo(w)ess normalization*. The simplest version of this assumes that the majority of the genes are non-differentially expressed and for this reason the log-ratios are expected to be centered around zero for all intensities. Under the above assumption curves estimated using robust local regression methods such as lowess [9, 10] or loess [11], or curves modeled by smoothing splines [15] will be good approximations of the $m_{r=1}(A)$ function, which then can be subtracted from the observed log-ratios

$$M \leftarrow M - m_{r=1}(A) = m_r(A) - m_{r=1}(A). \qquad (12)$$

Under an affine transform, $m_r(A)$ and $m_{r=1}(A)$ are as in Equation (9), but we do not know of a closed form expression for (12). An example of a curve-fit normalization under the affine transform is depicted in Figure 4. Note that the asymmetry between up- and down-regulated genes is *not* corrected for. Moreover, if we look at the overlaid $(\log_2 x_G, \log_2 x_R)$ grid in the left graph of Figure 4, we find that the curve-fit normalization warps it and removes the otherwise orthogonal relationship between $\log_2 x_R$ and $\log_2 x_G$ (if the $(2A, M)$ plane is considered).

4.2 Perpendicular translation normalization revisited

The (perpendicular) shift-log method proposed in [21] is a normalization method that corrects for differences in the channel biases. It normalizes the log-ratios us-

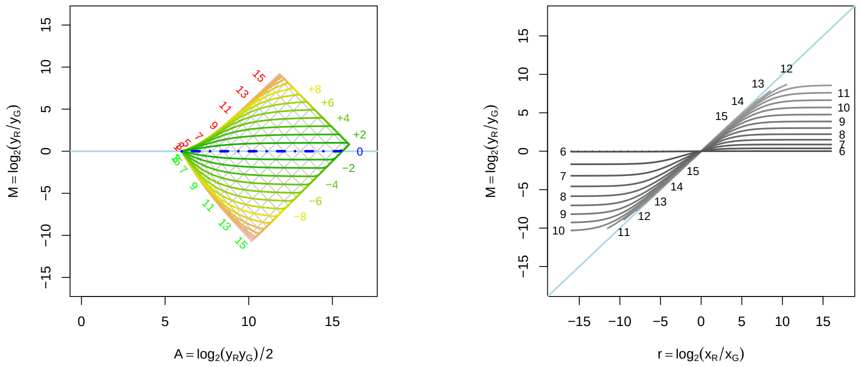


Figure 4: Curve-fit normalization of affinely transformed data where $(a_G, a_R) = (200, 20)$ and $(b_G, b_R) = (1.4, 0.8)$. *Left:* Log-ratios as a function of log-intensities for different fold changes. Note that the distance between up and down-regulated genes at any intensity is the same before and after the normalization. *Right:* Normalized log-ratios versus true log-ratios. We see that intensity-dependent artifacts have been removed for the observed and true log-ratios where all curves intersect (here at $(0, 0)$).

ing a translation transform where the signal in one channel is added by a constant and the signal in the other channel is subtracted by the same constant;

$$y_{R,i} \leftarrow a_R + b_R x_{R,i} + a \tag{13}$$

$$y_{G,i} \leftarrow a_G + b_G x_{G,i} - a \tag{14}$$

for all spots i. We refer to this translation normalization transform as the *perpendicular translation normalization*, because it moves data (x_G, x_R) perpendicular to the $x_R = x_G$ line. From Equation (10), we get that the observed log-ratios $m_r(A)$ can be made independent of the intensities if and only if

$$a = \frac{r b_R a_G - b_G a_R}{b_G + r b_R}, \ r > 0. \tag{15}$$

As this is a function of r, it is only for a single fold change at the time this method can make M independent of A. The most common choice is $r = 1$ for which the optimal perpendicular shift is

$$a = \frac{b_R a_G - b_G a_R}{b_G + b_R}, \tag{16}$$

which is the weighted difference between a_R and a_G with weights $b_G/(b_G + b_R)$ and $b_R/(b_G + b_R)$, respectively. The distance from the $r = 1$ curve to the $M = 0$ curve for the optimal perpendicular shift is $\log_2 b$. In other words, the perpendicular shift normalization will not remove an overall bias in the log-ratios (although it is not hard to estimate b afterward). The optimal shift for $(a_G, a_R) = (200, 20)$ and $(b_G, b_R) = (1.4, 0.8)$ is $a = 60$. The distance between the $r = 1$ and the $M = 0$ curve is 0.57. The result of this normalization is depicted in Figure 5. Note that $m_r(A)$ after normalization is constant for $r = 1$.

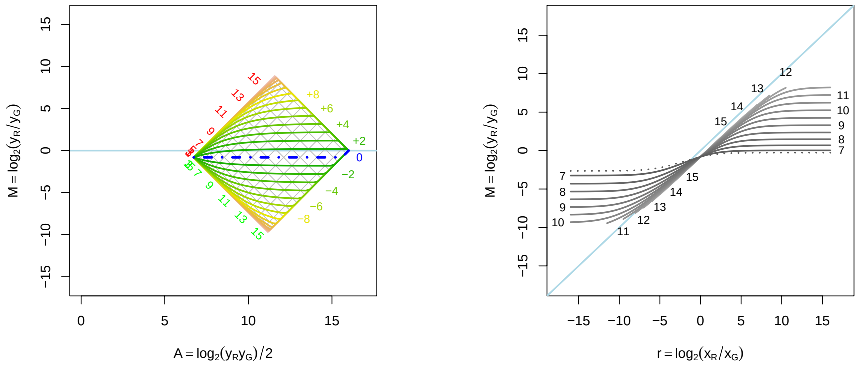


Figure 5: Perpendicular translation normalization of affinely transformed data where $(a_G, a_R) = (200, 20)$ and $(b_G, b_R) = (1.4, 0.8)$. The optimal amount of normalization shift in the raw data is $a = 60$, which corresponds to $a'_R = 80$ and $a'_G = 140$. *Left:* Log-ratios as a function of log-intensities for certain fold changes. The $r = 1$ curve (dot-dash blue) is horizontal, that is, for this specific value of r and a the log-ratios are independent of the log-intensities. *Right:* Normalized log-ratios versus true log-ratios. From this graph it is clear that we obtain the minimum error in log-ratios at zero-fold change. The dotted curves correspond to the minimum and maximum log-intensities possible to observe.

As suggested by [21], one way to find the optimal shift a is to minimize the curvature by minimizing the variation of the log-ratios after applying the shift a. To do this robustly, the median absolute deviation (MAD) can be used as measure of variation:

$$\hat{a} = \arg\min_{a} \text{MAD}_{1 \le i \le I} \left(M_i(a) \right). \tag{17}$$

A problem with the perpendicular translation normalization methods is that the optimal shift can result in non-positive signals making a huge number of expression ratios invalid. The normalization method discussed next does not have this problem, but on the other hand, it will not work or work badly under certain conditions.

4.3 Parallel translation normalization revisited

Another very similar normalization method that corrects for differences in biases in two channels is what we refer to as the *parallel translation normalization* method. For historical reasons, but also because it will contribute to our discussion about background correction, the shift-log method proposed by [26] for stabilizing (read decreasing or shrinking) the variance of the measured log-ratios is of interest. A side effect of their method is that it can correct for intensity-dependent curvature. It is based on a translation transform where the same constant is added to the signals in both channels;

$$y_R \leftarrow a_R + b_R x_R + a \tag{18}$$

$$y_G \leftarrow a_G + b_G x_G + a \tag{19}$$

where in their paper $a \geq 0$, but it is possible that a is negative. We refer to this special affine translation normalization as the *parallel translation normalization*, since it moves data (x_G, x_R) parallel to the $x_R = x_G$ line. Again, as this is a function of r, M can only be made independent of A for one unique r at the time. See Equation (10). For $r = 1$ the optimal parallel shift is

$$a = \frac{b_R a_G - b_G a_R}{b_G - b_R},\ b_G \neq b_R, \tag{20}$$

which can be estimated as in (17). For example, if $(a_G, a_R) = (200, 20)$ and $(b_G, b_R) = (1.4, 0.8)$, the optimal parallel shift is $a = 220$ with the $r = 1$ curve

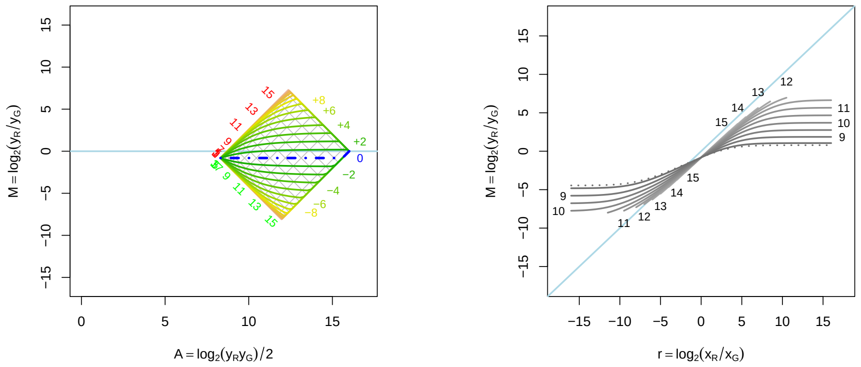


Figure 6: Parallel translation normalization of affinely transformed data where $(a_G, a_R) = (200, 20)$ and $(b_G, b_R) = (1.4, 0.8)$. The optimal amount of normalization shift in the raw data is $a = 220$, which corresponds to an effective shift of $(a'_G, a'_R) = (420, 240)$. *Left:* Log-ratios as a function of log-intensities for certain fold changes. The $r = 1$ curve (dot-dash blue) is horizontal, that is, for this specific value of r and a the log-ratios are independent of the log-intensities. *Right:* Normalized log-ratios versus true log-ratios. From this graph it is clear that we obtain the minimum error in log-ratios at zero-fold change.

0.57 units below the $M = 0$ line. The result of this normalization is depicted in Figure 6. From the above expression, we also see that an optimal value of a indeed can be negative. For example, if $(a_G, a_R) = (200, 140)$ and $(b_G, b_R) = (1.4, 0.8)$, the optimal parallel shift is $a = -60$, which corresponds to an effective shift of $(a'_G, a'_R) = (140, 80)$. However, it can also result in non-positive signals and therefore undefined log-ratios. For example, if $(a_G, a_R) = (200, 20)$ and $(b_G, b_R) = (1.4, 0.8)$, the optimal parallel shift is $a = -440$, which corresponds to an effective shift of $(a'_G, a'_R) = (-420, -240)$. Moreover, from (20) we see that when the scale parameters are equal there is no solution. For $r = 1$ the derivative is independent of a. This is because in such cases data is moved in parallel to the $x_R = x_G$ line making it impossible to get closer. As in the case of the perpendicular shift normalization, the distance between the $r = 1$ curve and the $M = 0$ curve is $\log_2 b$. Hence, the parallel shift normalization will not remove an overall bias in the log-ratios either and rescaling is necessary.

4.4 Single-channel translation normalization

The perpendicular shift normalization method has the problem that it can introduce non-positive signals and the parallel shift normalization method works badly or not at all under certain conditions. A hybrid of the two method is a normalization method that translates the signals in one of the channels at the time according to

$$y_R \leftarrow a_R + b_R x_R + a \cdot \mathbb{I}(a \geq 0) \tag{21}$$

$$y_G \leftarrow a_G + b_G x_G - a \cdot \mathbb{I}(a < 0), \tag{22}$$

where $\mathbb{I}(\cdot)$ is the indicator function. This will not generate non-positive signals as only positive translations are applied. Moreover, because only one channel is shifted an optimal shift will always be found. The same objective function as before can be used to find the optimal shift. However, the derivative of the objective function will be discontinuous at $a = 0$, but in practice this is not a problem and a can be estimated as done previously.

4.5 Rescale normalization

The above translation normalization methods removed the curvature by adjusting the bias parameters in $a_R - ba_G$ keeping the relative scale b fixed. Similarly, if we keep the bias parameters fixed, we can remove curvature by adjusting the relative scale b. We have shown that the scanner introduces *scale (PMT) insensitive* biases in both the red and the green channel [6]. Thus, by adjusting the PMT settings such that the curvature of the pre-scanned data is as small as possible one minimizes $|a_R - ba_G|$. Indeed, this strategy is in practice used by many. However, from above we know that this can equally well be done numerically. It is much more important to adjust the PMT (and laser) settings such that the dynamical range of the signals is as large as possible. Furthermore, as scanner settings are often adjusted for each array separately, there will be a discrepancy between arrays, which in any case has to be normalized for.

4.6 Dye-swap normalization revisited

The *dye-swap normalization*, also known as the *reverse labeling* and the *paired-slides normalization*, is a balanced experimental design for two-color microarrays that can be used whenever two technically replicated hybridizations are available.

Assume we have two different sets of cell populations, A and B, for which we would like to compare the relative gene expressions $r = x_A/x_B$ for all genes. After cDNA is obtained through reverse transcription the two samples are each split into two identical parts, one which is labeled with a red fluorescent dye and one which is labeled with a green fluorescent dye. We mix the red cDNA cocktail from the sample A with the green ditto from the sample B and let them co-hybridize to the DNA on the first array. After scanning we observe $(f_{G_1}(x_B), f_{R_1}(x_A))$. We do the same for the remaining red-green pair for which we obtain the observed expression levels $(f_{G_2}(x_A), f_{R_2}(x_B))$. Now, applying the dye-swap normalization suggested by [36] we get that

$$\begin{aligned} M = \frac{1}{2}(M_1 + M_2) &= \frac{1}{2}\left(\log_2 \frac{f_{R_1}(x_A)}{f_{G_1}(x_B)} - \log_2 \frac{f_{R_2}(x_B)}{f_{G_2}(x_A)}\right) \\ &= \frac{1}{2}\left(\log_2 \frac{f_{R_1}(x_A)}{f_{R_2}(x_B)} + \log_2 \frac{f_{G_2}(x_A)}{f_{G_1}(x_B)}\right) = \frac{1}{2}(M'_1 + M'_2) \end{aligned} \tag{23}$$

and similarly for the log-intensities

$$\begin{aligned} A = \frac{1}{2}(A_1 + A_2) &= \frac{1}{4}(\log_2 [f_{R_1}(x_A) f_{G_1}(x_B)] + \log_2 [f_{R_2}(x_B) f_{G_2}(x_A)]) \\ &= \frac{1}{4}(\log_2 [f_{R_1}(x_A) f_{R_2}(x_B)] + \log_2 [f_{G_2}(x_B) f_{G_2}(x_A)]) = \frac{1}{2}(A'_1 + A'_2). \end{aligned} \tag{24}$$

Thus, the result of a dye-swap can be written as the average of two "virtual" hybridizations (A'_1, M'_1) and (A'_2, M'_2). Moreover, if (and only if) the measurement functions are equal for each array, that is, $f_{R_1}(\cdot) = f_{R_2}(\cdot)$ and $f_{G_1}(\cdot) = f_{G_2}(\cdot)$, then the observed ratios will be identical to the true ratios *for non-differentially expressed genes*. For this to be true for differentially expressed genes we already know that they also have to be linear, that is, affine with zero intercept.

Several authors [25, 22] have reported that dye-swap normalization does remove curvature, but less successful results have also been reported [33]. To better understand the reasons why and when dye-swap normalization works and not, we dissect the measurement functions $f_c(\cdot)$ of the four channels $c = R_1, G_1, R_2, G_2$ into $(v_c \circ u_c \circ t_c \circ s_c)(\cdot)$ where $s_c(\cdot)$ models the process of all steps up to the step where the (not yet labeled) cDNA sample is obtained, $t_c(\cdot)$ models the labeling, and $u_c(\cdot)$ models the following steps including the hybridization, and $v_c(\cdot)$

models the scanning etc. As channel R_1 and G_2 are from sample A and the other two are from sample B, we know that $s_{R_1}(\cdot) = s_{G_2}(\cdot) = s_A(\cdot)$ and $s_{R_2}(\cdot) = s_{G_1}(\cdot) = s_B(\cdot)$. Furthermore, if the labeling process is well controlled we can assume that $t_{R_1}(\cdot) \approx t_{R_2}(\cdot) \approx t_R(\cdot)$ and $t_{G_1}(\cdot) \approx t_{G_2}(\cdot) \approx t_G(\cdot)$. When hybridizing channel 1 and 3 to array one and the other two to array 2 we approximately have that $u_{R_1}(\cdot) \approx u_{G_1}(\cdot) \approx u_1(\cdot)$ and $u_{R_2}(\cdot) \approx u_{G_2}(\cdot) \approx u_2(\cdot)$. The properties of the glass arrays can be incorporated into these two measurement functions. Moreover, if we keep the same scanner settings for both arrays and assume everything else equal, we have that $v_{R_1}(\cdot) \approx v_{R_2}(\cdot) \approx v_R(\cdot)$ and $v_{G_1}(\cdot) \approx v_{G_2}(\cdot) \approx v_G(\cdot)$. The overall measurement function for the channels are then approximately

$$f_{R_1} = v_R \circ u_1 \circ t_R \circ s_A \tag{25}$$

$$f_{G_1} = v_G \circ u_1 \circ t_G \circ s_B \tag{26}$$

$$f_{R_2} = v_R \circ u_2 \circ t_R \circ s_B \tag{27}$$

$$f_{G_2} = v_G \circ u_2 \circ t_G \circ s_A. \tag{28}$$

For the dye-swap normalization to be efficient, we conclude that we must control the process of extracting the RNA etc. to an extent such that we can expect $s_A(\cdot) \approx s_B(\cdot)$. Moreover, we must also be able to reproduce hybridizations well on high quality arrays such that $u_1(\cdot) \approx u_2(\cdot)$. If these requirements are met, then any differences in physical properties of the dyes or the red and the green channels in general will be self-normalized.

Turning to the affine model, from Equation (23) we get that a dye-swap normalization of affinely transformation data will result in the two virtual hybridizations

$$M_1' = \log_2 \frac{a_R + b_R x_A}{a_R + b_R x_B},\ M_2' = \log_2 \frac{a_G + b_G x_A}{a_G + b_G x_B}, \tag{29}$$

and similar for A_1' and A_2'. For both of them, the signals in both channels have undergone identical affine transformations. We know from before that identical transformation in both channels does not introduce curvature for the non-differentially expressed genes and that there is a symmetry between up- and down-regulated genes, cf. perpendicular and parallel shift normalization. Moreover, optimally there is no bias and the above simplifies to $M = \log_2 (x_A/x_B)$ and $A = \frac{1}{2} \log_2 (x_A x_B)$ as in [36], which *are* the true expression levels. If the biases in any of the two replicated channels are not equal ($a_{R_1} \neq a_{R_2}$), the dye-swap normalization will not work.

The above discussion assumed that the same cell samples have been replicated. If biological replicates are used, an additional source of variability is introduced. However, as long as we can assume that for most genes $x_{A_1} \approx x_{A_2}$ and $x_{B_1} \approx x_{B_2}$, dye-swap normalization should still perform well.

4.7 Alternative dye-swap normalization

An alternative to the above dye-swap normalization method is to average the observed expression levels *before* taking the logarithm

$$M = \log_2 \frac{(f_{R_1}(x_A) + f_{G_2}(x_A))/2}{(f_{R_2}(x_B) + f_{G_1}(x_B))/2} = \log_2 \frac{f_{R_1}(x_A) + f_{G_2}(x_A)}{f_{R_2}(x_B) + f_{G_1}(x_B),} \tag{30}$$

and analogously for A. This approach uses the *arithmetic mean* of the observed signals whereas the previous dye-swap method used the *geometric mean*. To be able to say more about the difference between the two approaches, we turn to the affine transformation for which we have

$$M = \log_2 \frac{(a_R + a_G) + (b_R + b_G)x_A}{(a_R + a_G) + (b_R + b_G)x_B} \tag{31}$$

$$A = \frac{1}{2} \log_2 \left[\{(a_R + a_G) + (b_R + b_G)x_A\}\{(a_R + a_G) + (b_R + b_G)x_B\}.\right] \tag{32}$$

Again, we note that the dye-swap method makes the transforms in the resulting two virtual channels equal. However, comparing the bias in log-intensities between the geometrical and the arithmetical approaches, we see that for the latter we have

$$A_0 = \frac{1}{2} \log_2 \left(\frac{1}{4}(a_R + a_G)(a_R + a_G) \right) = \log_2 \frac{a_R + a_G}{2} \tag{33}$$

whereas for the former we have

$$A_0 = \frac{1}{2}(A'_{1,0} + A'_{2,0}) = \frac{1}{2} \left(\frac{1}{2} \log_2 a_R^2 + \frac{1}{2} \log_2 a_G^2 \right) = \log_2 \sqrt{a_R a_G}. \tag{34}$$

Because $(a_R + a_G)/2 \geq \sqrt{a_R a_G}$, we conclude that A_0 is always larger for *arithmetic dye-swap* than for the geometric one, and from previous results we know that the larger A_0 is the more curvature we have for the differentially expressed genes.

However, there are other differences too. For instance, if each microarray glass array (the $u_c(\cdot)$ functions above) introduces the same additive bias to both channels and this bias is different between arrays, but otherwise everything else is the same, that is, $a_{R_2} = a_{R_1} + a$ and $a_{G_2} = a_{G_1} + a$, then the arithmetic dye-swap would do better

$$M_a = \log_2 \frac{(a_{R_1} + a_{G_1} + a) + (b_R + b_G)x_A}{(a_{R_1} + a + a_{G_1}) + (b_R + b_G)x_B} \tag{35}$$

than the *geometric dye-swap*

$$M_g = \frac{1}{2}\left(\log_2 \frac{a_{R_1} + b_R x_A}{(a_{R_1} + a) + b_R x_B} + \log_2 \frac{(a_{G_1} + a) + b_G x_A}{a_{G_1} + b_G x_B}\right), \tag{36}$$

which in such case fails to remove curvature.

4.8 Two-channel quantile normalization

Two-channel, or in general multi-channel *quantile normalization* [7, 37] is based on and relies on the *assumption that the true gene-expression levels in the two biological samples are approximately equally distributed.* If the measurement functions in the two channels, say $f_R(\cdot)$ and $f_G(\cdot)$, are different, then the distributions of the measured signals in the two channels are different even if the underlying distributions of the true expression levels would be identical. By estimating the distributions of the two channels and making them equal, for instance to an average distribution, the log-ratios for the *non-differentially* expressed genes will be unbiased and independent of the intensities. Thus, making the density functions of the measured data equal for the two channels is the same as making their transformation functions equal, say to $f_{RG}(\cdot)$, and equal transformation functions will make M independent of A for the non-differentially expressed genes. If $f_{RG}(\cdot)$ could be made linear too, this would be true for all fold changes.

If we turn to the affine transform, two-channel quantile normalization succeeds to remove intensity-dependent effects, because the bias a_R and a_G in the two channels are made identical when the quantiles of the signals are made the same. The elimination of the bias in log-ratios is because b_R are made equal to b_G. Hence, two-channel quantile normalization can be considered to be both a method that corrects for differences in bias between two channels, but also a method that corrects for biases in the expression ratios. In Figure 7, the two-channel quantile normalization of affinely transformed data where $(a_G, a_R) = (200, 20)$ and $(b_G, b_R) = (1.4, 0.8)$ is depicted. To the left are the affine transformations for the red and the green channels shown. The (upside-down) curve at the bottom shows a hypothetical density function of the true (log) gene-expression levels expected to be equal in both samples. The distributions of the affinely transformed signals are shown in the (rotated) density functions at the left (red and green curves). The average signal density (middle gray curve) to be normalized toward corresponds to a common measurement function (gray function in the main plot). The two-channel quantile normalized data is shown in the M versus A plot to the right. We see that the curvature of the non-differentially expressed genes is removed.

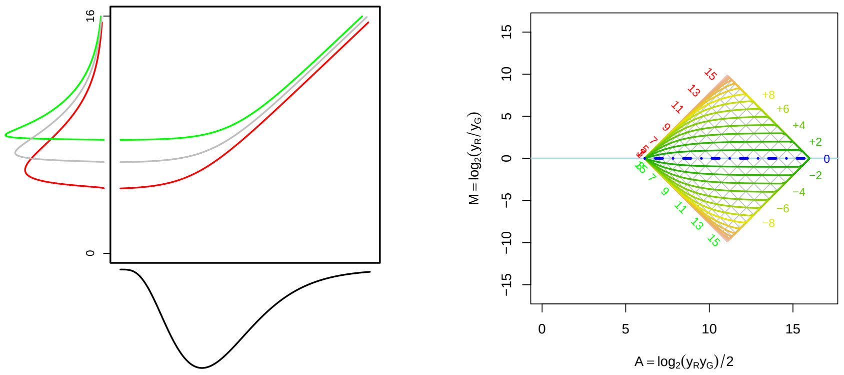


Figure 7: Equalizing the signal densities of the two channels remove the intensity dependency of the log-ratios of non-differentially expressed genes. *Left:* Equal gene-expression distributions in both channels will under the non-channel balanced affine transform where $(a_G, a_R) = (200, 20)$ and $(b_G, b_R) = (1.4, 0.8)$ turn into two different densities for the measured data. *Right:* Normalizing the non-equal densities of the two channels makes the log-ratios of the non-differentially expressed genes zero for all intensities. All axes are logged.

4.9 Background subtraction as a normalization or calibration method

We have observed that the log-ratios of the *background signals* show the same intensity-dependent effects as the *foreground signals*, which suggests that the background signals undergo the same transformation as the foreground signals. An example of this is shown in Figure 8, where the background and the foreground estimates from one hybridization are plotted in the same M versus A scatter plots. This may be explained by the results in [6]. A widely adopted rationale for background correction is the assumption that the region that defines the spot is contaminated with the same physical noise that can be observed in the surrounding regions. Background noise is commonly explained to be due to dust particles, DNA contaminated buffers, failed washing during printing or hybridization, cross hybridization etc. [23, 29]. This type of background noise is often assumed to add to the foreground signal. Thus, to obtain the true signal the background is subtracted from the foreground signal

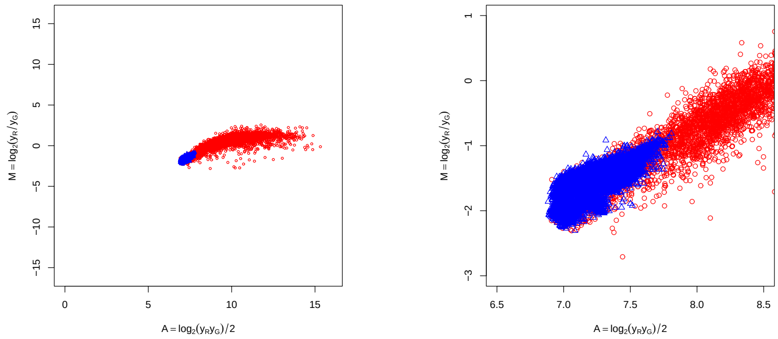


Figure 8: Transformation of background signal. *Left:* An M versus A scatter plot where background signals (blue triangles) and foreground signals (red circles) lye along the same curve, which is evidence that both have been transformed identically. *Right:* A zoom-in of the left graph. Data is from [20].

$$y_{c,i} \leftarrow y_{c,i}^{(\text{fg})} - y_{c,i}^{(\text{bg})} \tag{37}$$

where $y_{c,i}^{(\text{fg})}$ is the estimated foreground signal and $y_{c,i}^{(\text{bg})}$ is the estimated background signal for channel c and spot i. Under a transformation that is dominated by an affine function for lower intensities (of the same order as the background), subtracting the background from the foreground will shift the biases toward zero and background subtracted signals will have less curvature in the (A, M) plane than non-background subtracted signals (not shown). In this sense we can consider background subtraction to be a normalization (or a calibration) method. However, just because the log-ratios as a function of the log-intensities becomes more flat, it does not mean that the foreground estimates are contaminated by the same signal that is found in the background area (commonly the regions surrounding the spots). Instead, the explanation could be that the background estimates happen to be approximately equal to the amount of shift in the foreground signals. Different image analysis software estimate the background signal differently based on different algorithms such as fixed-size circles, adaptive circles, morphological estimates, and pixel intensity distributions. Although comparative studies have been conducted [34, 3], it is still not clear which background estimate is most correct. Some methods give higher background estimates than others, which means that they all correct for channel biases by different amounts, which is another argument for why we most likely do have biases in our signals. Moreover, subtracting the background from the foreground signals commonly results in non-positive signals, which makes data harder to analyze, but also to interpret. Even though the background estimates were known to be unbiased, it is not surprising to get non-positive signals under the assumption that the foreground region is contaminated by the same noise as found in the background region. This is because both the foreground and the background signals are *estimates*. A pioneering work that makes use of this is [23], which emphasizes that the true signal can *not* be negative and uses a Bayesian approach to correct for this.

4.10 Result of a (relative) negative translation

If, for instance, too much background is subtracted or a threshold has to be passed before the reverse transcription takes place, one can imagine that $a_G, a_R < 0$. With negative shift the curves in Figure 2 bend downward instead of toward the left. Negative bias also applies if the observed signals are compared, not to the true signals, but to the signals obtained by another measuring technique that has a higher bias. Examples of such comparisons can be two-color microarray data compared to oligonucleotide (Affymetrix) data or two-color microarray data compared to QRT-PCR data. Negative bias can also be observed when control clones, spike-ins, negative and positive controls etc. are compared with the genes/ESTs of interest. The effect of a negative translation is depicted in Figure 9. The fan-out effect in the fold-change curves for the lower intensities is due to the negative translation. Note that this should not be mistaken for the fan-out effect due to decreasing signal-to-noise levels.

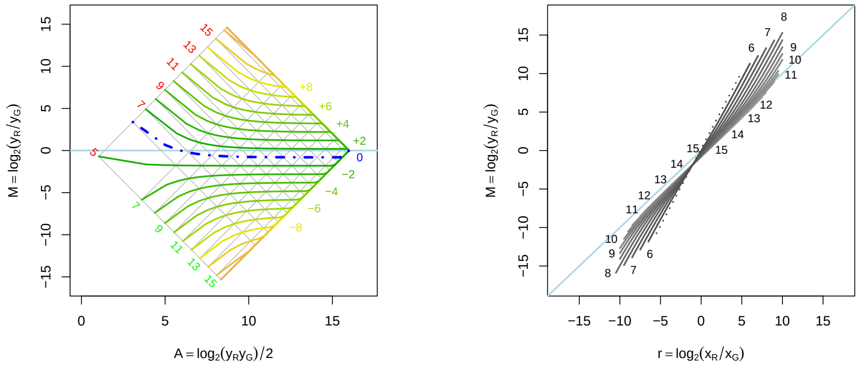


Figure 9: Affine transformation of the red and the green signals with negative translation where $(a_G, a_R) = (-87, -24)$, $(b_G, b_R) = (1.4, 0.8)$. *Left:* Log-ratios as a function of log-intensities for certain fold changes. *Right:* Translated log-ratios versus true log-ratios. The slope of a line fitted in the M versus M plot will be *larger* than one, which is due to the negative translation. Note that we have truncated both fold-change curves and the grid in the left graph and the intensity curves in the right graph such that $x_R, x_G \geq 1$.

4.11 Robust affine normalization

It is clear that it is essential to correct for biases when two or multi-channel cDNA microarray data is normalized. We can obtain estimates of a_R, a_G and b as follows. Define $\alpha = a_R - ba_G$ and $\beta = b = b_R/b_G$. For non-differentially expressed genes (without noise) we have that

$$y_{R,i} = \alpha + \beta y_{G,i}; \; i = 1, 2, \ldots, I. \tag{38}$$

Define $\mathbf{y} = \{\mathbf{y}_i\}_{i=1}^{I}$ where $\mathbf{y}_i = (y_{G,i}, y_{R,i})$ and let

$$Q(\alpha, \beta; \mathbf{y}) = \sum_{i=1}^{I} w_i r_i(\alpha, \beta; \mathbf{y}_i)^2 \tag{39}$$

be our objective function where $r_i(\alpha, \beta; \mathbf{y}_i) > 0$ is the orthogonal Euclidean distance between $\mathbf{y}_i$ and the line $L(\alpha, \beta)$ with intercept α and slope β. The estimate of α and β is then

$$(\hat{\alpha}, \hat{\beta}) = \arg \min_{(\alpha,\beta)} Q(\alpha, \beta; \mathbf{y}). \tag{40}$$

With $w_i = 1$ for all observations we obtain standard principal component analysis (PCA), which minimizes the orthogonal distances in the L_2 norm. With weights $w_i = 1/(r_i(\hat{\alpha}, \hat{\beta}; \mathbf{y}_i) + \delta)$ we can minimize the distances in the L_1 norm, if we let $\delta \to 0^+$. Other robust estimators can be obtained by choosing other weight functions, but we choose to optimize in L_1. Moreover, if one suspects a non-symmetric distribution of data points around the line, then a trimmed version of the weight function should be considered. In practice, the above optimization can be done by an *iterative reweighted principal component analysis* (*IWPCA*) scheme. For iteration $l = 1, 2, \ldots$, minimize (39) using weighted PCA where $w_i^{(1)} = 1$ and $w_i^{(l+1)} = 1/(r_i(\alpha^{(l)}, \beta^{(l)}; \mathbf{y}_i) + \delta)$ with δ being a small positive number to avoid infinite weights.

As a last step, in order to get estimates of the three parameters a_R, a_G and b from the two parameter estimates $\hat{\alpha}$ and $\hat{\beta}$, we introduce an additional constraint. Let $y_{c,\min} = \min_i y_{c,i}$ for $c = \{R, G\}$. We then choose

$$\hat{b} = \hat{\beta} \tag{41}$$

$$\hat{a}_G = \max\{a_G; a_G < y_{G,\min} \wedge \hat{\alpha} + \hat{b}a_G < y_{R,\min}\} \tag{42}$$

$$\hat{a}_R = \hat{\alpha} + \hat{b}\hat{a}_G \tag{43}$$

to be the estimates of the bias and the scale parameters in model (3). This constraint is only correct in the error free model. If we allow noise, say

$$y_{R,i} = a_R + b_R x_{R,i} + \varepsilon_{R,i} \tag{44}$$

$$y_{G,i} = a_G + b_G x_{G,i} + \varepsilon_{G,i}, \tag{45}$$

where $E[\varepsilon_{c,i}] = 0$ and $V[\varepsilon_{c,i}] = \sigma_{c,i}^2$ for $c = \{R, G\}$, it is possible that the bias terms a_R and a_G are indeed larger than the smallest observed value in the respective channel. This is especially important if the distributions of $\varepsilon_{c,i}$ for $c = \{R, G\}$ have heavy negative tails. However, as already mentioned, we do not know enough about the various noise sources, and to avoid the problem with non-positive signals, we have decided to keep the above constraint. We are aware of the fact that the estimates in such cases are inconsistent, but we believe that the error we do by subtracting as much bias as possible without introducing non-positive signals, as in (41)-(43), is much smaller than not correcting for the bias at all. An alternative, which introduces negative estimates, is to replace $y_{c,\min}$ in (42) with $y_{c,(i)}$, where order index (i) is chosen such that i non-positive signals are obtained in channel c.

Furthermore, it has been observed that the noise in each channel is roughly proportional to the signal strength, that is, $\sigma_{c,i} \propto x_{c,i}$. Thus, a positive side effect of the above estimation algorithm is that, contrary to have equal weights for all spots ($w_i = 1$), more weight will be given to the low-intensity spots compared to the high-intensity ones. This will make the method more robust to saturation and other non-linear effects that might occur at high intensities, effects for which classical line fits, which rely on homoscedasticity, would fail.

Finally, with backward transformation (4) based on estimates for $(\hat{a}_R, \hat{a}_G, \hat{b})$, we translate and rotate data such that it falls around the diagonal line that goes through $(0, 0)$ and $(1, 1)$. In Figure 10 an example is given where the above affine normalization method has been applied to a two-color microarray data set. It is clear that the curvature in the M versus A scatter plot is removed by the normalization method. For a comparison of the bias in the log-ratios, see also Figure 3. Data is the same as in [6] and a more detailed analysis of differentially expressed genes etc. will be published elsewhere.

Furthermore, if more than one array is normalized, we suggest that the signals from all arrays and both channels are normalized together. For this to be realistic, the assumption that most genes are non-differentially expressed for *all* possible hybridization/channel pairs must be added. However, this is not a problem since the experimental design is often set up such that this is true. For instance, in two-channel microarrays experiments it is common to hybridize one test sample

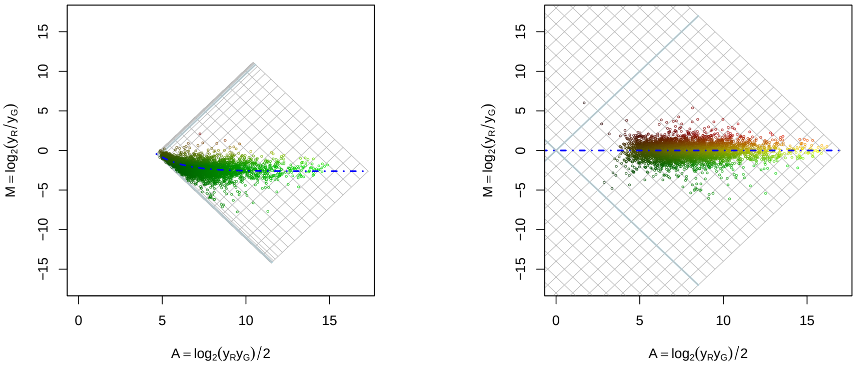


Figure 10: Log-ratios versus log-intensities before (left) and after (right) a robust affine normalization. The intensity-dependent trend of the log-ratios in the raw data is removed by the affine normalization. The affine parameters were estimated to $(a_R, a_G, b) = (21.5, 29.3, 0.163)$, which for instance gives that $(A_0, M_0) = (4.65, -0.45)$ and $M_\infty = -2.62$.

and one reference sample, which is selected such that it does not differ too much from the test sample, to the same array. The same reference is then used between arrays (in either channel). Thus, since each test-reference pair is "close" to each other, all test-test pairs should approximately be "close" to each other also.

The multi-channel algorithm can be summarized as follows. Say there are N arrays each hybridized with two $K = 2$ samples (colors). The observations $\mathbf{y}_i = (y_{R_1,i}, \ldots, y_{R_N,i}, y_{G_1,i}, \ldots, y_{G_N,i})$ for genes $i = 1, 2, \ldots, I$ then span a KN-dimensional space. Analogously to the above two-dimensional procedure, we can fit a robust line L through data in $\mathbb{R}^{KN}$ and constrain the estimate of $\mathbf{a} = (a_{R_1}, \ldots, a_{R_N}, a_{G_1}, \ldots, a_{G_N})$ by enforcing that $\mathbf{a} \prec \mathbf{y}_i; \forall i$ where $\prec$ is the component-wise inequality. Backward transformation will again translate and rotate data such that it lies along the diagonal line. Moreover, normalizing all arrays at once will bring the signals from all hybridizations onto the same scale and no further, so called, between-slide scale normalization is needed. This multi-dimensional affine normalization method with slightly different constraints was successfully used in a multiscan calibration two-color microarray experiment [6].

An implementation of the above algorithm is made available in the R [31, 18] package named *aroma* [5] (formerly known as the com.braju.sma package). The only parameter in the algorithm that has to be specified is δ, However, its value is not critical and we have found that for instance $\delta = 0.02$ works well in general and is therefore the default value. Thus, in practice the algorithm is completely automatic. For instance, `normalizeAffine(rg)` will normalize all arrays and all channels in the microarray object `rg` at once. Moreover, the method can be applied to any subsets of genes separately such as print-tip groups, clone groups and spike-ins.

5 Discussion

In previous sections, we did not discussed the variance stabilizing methods described by [17, 14], which are based on error models that also contain channel-specific bias terms. These bias terms are estimated and corrected for. Thus, those methods do indeed correct for intensity-dependent effects. Because they are based on specific error models, but also because they stabilize the log-ratio variances, they do not fit well into the above deterministic discussion. However, we do believe that the directions drawn up by their underlying error models are promising.

If we compare the robust affine normalization method in Section 4.11 with for instance the perpendicular and the parallel translation normalization methods optimized by minimizing the curvature, we find that there are similarities, because minimizing the curvature is identical to finding estimates of the bias parameters along the line $L(\alpha, \beta; \mathbf{y})$. Assuming a pure affine transformation, there are also similarities to the curve-fit method, which fit approximately the same line (curve) through data. The only difference then is how data is transformed to meet the assumptions. The affine method translates and rescales data in the original domain whereas the curve-fit method operates in a rotated and log-transformed domain.

Moreover, the translation and the curve-fit methods rely on two-dimensional data (log-ratios) and it is not clear how to generalize them to multi-dimensional data, although re-iterative versions such as cyclic loess [7] and the (multi-dimensional) contrast based method [1] have been suggested. Our affine normalization method is not limited to two-dimensional data, but can be applied to any number of channels, which means that for instance three and four-color microarray data can be normalized as easily as two-color data.

It is also interesting to note the close relationship between the quantile and affine normalization method. In the case of an affine transformation, both will fit roughly the same line (curve) through data and translate it into the original domain. The difference is how data is normalized to this line. In quantile normalization the data points are shifted such that the sample densities of both channels are made identical. This results in new measurement functions, which are not necessarily linear, but still such that for non-differentially expressed genes the ratios between them are one. However, the affine normalization model can be though of as a quantile normalization method with special constraints on the underlying densities. An interesting continuation of the affine model and quantile normalization method would be how to relax the affine constraint by using other parametric or semi-parametric models. One possibility is to add smoothness constraints to the transformation functions using smoothing splines [15], which then also would provide a probabilistic framework. Moreover, in the spirit of [23], it would be interesting to incorporate an empirical Bayes component to “allow” non-positive signals too.

Acknowledgments

I (Henrik) am greatly thankful to Professor Terry Speed for taken me on as a visiting student twice. If it would not be for his challenging questions and great discussions with him and his enthusiastic colleagues and friends at both UC Berkeley and at Walter and Eliza Hall Institute of Medical in Melbourne (who are too many to mention by name but you all know who you are), this work would not have been done. I am also in great debt to Patyaksha (Asa) Wirapati (formerly at WEHI in Melbourne) who help me identify the core of what I was trying to do when I first started to do simulations of affine models in 2002. Thanks to Gordon Smyth for initial help on calculus. Many thanks to Halfdan Grage for fruitful discussions on PCA. My research visits, and thus this work, have been supported by The Swedish Foundation for International Cooperation in Research and Higher Education (STINT), The Fulbright Commission, The Foundation

Blanceflor Boncompagni-Ludovisi née Bildt, The Royal Swedish Academy of Sciences, and The Royal Physiographic Society in Lund. Finally, the microarray data from [6] was kindly provided by the SWEGENE DNA Microarray Resource Center at the BioMedical Center B10 in Lund, supported by the Knut and Alice Wallenberg foundation through the SWEGENE consortium.

References

[1] Magnus Åstrand. Contrast normalization of oligonucleotide arrays. *Journal of Computational Biology*, 10(1):95–102, 2003.

[2] Andrea Barczak, Madeleine Willkom Rodriguez, Kristina Hanspers, Laura L. Koth, Yu Chuan Tai, Benjamin M. Bolstad, Terence P. Speed, and David J. Erle. Spotted long oligonucleotide arrays for human gene expression analysis. *Genome Research*, 13(7):1775–85, Jul 2003.

[3] Anders Bengtsson. Microarray image analysis: Background estimation using region and filtering techniques. Master's Theses in Mathematical Sciences, Mathematical Statistics, Centre for Mathematical Sciences, Lund Institute of Technology, Sweden, December 2003. 2003:E40.

[4] Henrik Bengtsson. Identification and normalization of plate effects in cDNA microarray data. Preprints in Mathematical Sciences 2002:28, Mathematical Statistics, Centre for Mathematical Sciences, Lund University, Sweden, 2002.

[5] Henrik Bengtsson. aroma - An R Object-oriented Microarray Analysis environment. Preprint in Mathematical Sciences 2004:18, Mathematical Statistics, Centre for Mathematical Sciences, Lund University, Sweden, 2004.

[6] Henrik Bengtsson, Göran Jönsson, and Johan Vallon-Christersson. Calibration and assessment of channel-specific biases in microarray data with extended dynamical range. Preprints in Mathematical Sciences 2003:37, Mathematical Statistics, Centre for Mathematical Sciences, Lund University, Sweden, 2003. Submitted.

[7] B.M. Bolstad, R.A. Irizarry, M. Åstrand, and T.P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–93, Jan 2003.

[8] M. Callow, S. Dudoit, E. Gong, T. Speed, and E. Rubin. Microarray expression profiling identifies genes with altered expression in HDL-deficient mice. *Genome Research*, 10(12):2022–9, December 2000.

[9] W.S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of American Statistics Association*, 74:829–836, 1979.

[10] W.S. Cleveland. LOWESS: A program for smoothing scatterplots by robust locally weighted regression. *The American Statistician*, 35:54, 1981.

[11] W.S. Cleveland, E. Grosse, and W.M. Shyu. *Local regression models*. MIT Press/McGraw-Hill, 1992.

[12] Xiangqin Cui, M. Kathleen Kerr, and Gary A. Churchill. Data transformations for cDNA microarray data. Technical report, The Jackson Laboratory, USA, 2002.

[13] David J. Duggan, Michael Bittner, Yidong Chen, and Paul Meltzer & Jeffrey M. Trent. Expression profiling using cDNA microarrays. *Nature Genetics*, 21(1 Supplement):10–14, January 1999.

[14] B.P. Durbin, J.S. Hardin, D.M. Hawkins, and D.M. Rocke. A variance-stabilizing transformation for gene-expression microarray data. *Bioinformatics*, 18:S105–S110, 2002.

[15] P.J. Green and B.W. Silverman. *Nonparametric Regression and Generalized Linear Models - A roughness penalty approach*. Chapman and Hall, 1994.

[16] Shawn Handran, Chang Wang, and David Aziz. Assessing slide flatness, August 2001.

[17] Wolfgang Huber, Anja von Heydebreck, Holger Sültmann, Annemarie Poustka, and Martin Vingron. Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 1:1–9, 2002.

[18] Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

[19] Burble Industries Inc. *Photomultiplier Handbook*, 1980.

[20] A. Jögi, J. Vallon-Christersson, L. Holmquist, H. Axelson, Å. Borg, and S. Påhlman. Human neuroblastoma cells exposed to hypoxia: induction of genes associated with growth, survival and aggressive behavior. *Experimental Cell Research*, (accepted), 2004.

[21] M. Kathleen Kerr, Cynthia A. Afshari, Lee Bennett, Pierre Bushel, Jeanelle Martinez, Nigel J. Walker, and Gary A. Churchill. Statistical analysis of a gene expression microarray experiment with replication. Technical report, The Jackson Laboratory, Bar Harbor, Maine, 2001.

[22] M. Kathleen Kerr, M. Martin, and Gary A. Churchill. Analysis of variance for gene expression microarray data. *Journal of Computational Biology*, 7:819–837, 2000.

[23] Charles Kooperberg, Thomas G. Fazzio, Jeffrey J. Delrow, and Thoshio Tsukiyama. Improved background correction for spotted DNA microarrays. *Journal of Computational Biology*, 9:55–66, 2002.

[24] Xinmin Li, Weikuan Gu, Subburaman Mohan, and David J. Baylink. DNA microarrays: their use and misuse. *Microcirculation*, 9(1):13–22, Jan 2002.

[25] Matthew J. Marton, Joseph L. DeRisi, Holly A. Bennett, Vishwanath R. Iyer, Michael R. Meyer, Christopher J. Roberts, Roland Stoughton, Julja Burchard, David Slade, Hongyue Dai, Douglas E. Bassett Jr., Leland H. Hartwell, Patrick O. Brown, and Stephen H. Friend. Drug validation and identification of secondary drug target effects using DNA microarrays. *Nature Medicine*, 4(11):1293–1301, 1998.

[26] M. A. Newton, C. M. Kendziorski, C. S. Richmond, Frederick R. Blattner, and K. W. Tsui. On differential variability of expression ratios: Improving statistical inference about gene expression changes from microarray data. *Journal of Computational Biology*, 8:37–52, 2001.

[27] Latha Ramdas, Kevin R. Coombes, Keith Baggerly, Lynne Abruzzo, W. Edward Highsmith, Tammy Krogmann, Stanley R. Hamilton, and Wei Zhang. Sources of nonlinearity in cDNA microarray expression measurements. *Genome Biology*, 2(11):research0047.1–0047.7, 2001.

[28] David M. Rocke and Blythe Durbin. A model for measurement error for gene expression arrays. *Journal of Computational Biology*, 8(6):557–569, 2001.

[29] Mark Schena. *Microarrays Analysis*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2003.

[30] Mark Schena, Dari Shalon, Ronald W. Davis, and Patrick O. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270(5235):467–470, October 1995.

[31] R Development Core Team. R Language Definition (v1.7.0, draft), April 2003.

[32] Natalie Thorne. Personal communication. Details to appear in Thorne's PhD thesis., 2003.

[33] George C. Tseng, Min-Kyu Oh, Lars Rohlin, James C. Liao, and Wing Hung Wong. Issues in cDNA microarray analysis: quality filtering, channel normalization, models of variations and assessment of gene effects. *Nucleic Acids Research*, 29(12):2549–2557, Jun 2001.

[34] Yee Hwa Yang, Michael Buckley, Sandrine Dudoit, and Terry Speed. Comparison of methods for image analysis on cDNA microarray data. Technical Report 584, Department of Statistics, University of California at Berkeley, Nov 2000.

[35] Yee Hwa Yang, Sandrine Dudoit, Percy Luu, David M. Lin, Vivian Peng, John Ngai, and Terence P. Speed. Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucelic Acids Research*, 30(4):e15, Feb 2002.

[36] Yee Hwa Yang, Sandrine Dudoit, Percy Luu, and Terence P. Speed. Normalization for cDNA microarray data. Technical Report 589, Department of Statistics, University of California at Berkeley, 2000.

[37] Yee Hwa Yang and Natalie P. Thorne. Normalization for two-color cDNA microarray data. In Darlene R. Goldstein, editor, *Science and Statistics: A Festschrift for Terry Speed*, volume 40 of *Monograph Series*, pages 403–418. IMS Lecture Notes, 2003.

[38] H Yue, PS Eastman, BB Wang, J Minor, MH Doctolero, RL Nuttall, R Stack, JW Becker, JR Montgomery, M Vainer, and R Johnston. An evaluation of the performance of cDNA microarrays for detecting changes in global mRNA expression. *Nucelic Acids Research*, 29(8):E41–1, Apr 2001.

[39] Tony Yuen, Elisa Wurmbach, Robert L. Pfeffer, Barbara J. Ebersole, and Stuart C. Sealfon. Accuracy and calibration of commercial oligonucleotide and custom cDNA microarrays. *Nucleic Acids Research*, 30, 2003.

A Detailed calculations on the affine transform

A.1 x_G and rx_G as a function of A

We want to find an expression for $x_G = a^{-1}_{G,r}(A)$. First, doing the variable substitution $x_g = b_G x_G$ we have

$$A = \frac{1}{2}\log_2\left[(a_R + rbx_g)(a_G + x_g)\right] \tag{46}$$

where $b = b_R/b_G$ is the relative scale and $r = x_R/x_G$ is the true fold change. From this expression we can calculate the expression for $x_g = a^{-1}_{x_g,r}(A)$ as follows

$$\begin{aligned}
(a_R + rbx_g)(a_G + x_g) &= 2^{2A} \\
\iff rbx_g^2 + (a_R + rba_G)\,x_g + a_R a_G - 2^{2A} &= 0 \\
\iff x_g^2 + \left((rb)^{-1}a_R + a_G\right)x_g + (rb)^{-1}(a_R a_G - 2^{2A}) &= 0 \\
\iff x_g = -\frac{1}{2}\left((rb)^{-1}a_R + a_G\right) & \\
\pm\sqrt{\frac{1}{4}\left((rb)^{-1}a_R + a_G\right)^2 - (rb)^{-1}(a_R a_G + 2^{2A})} & \\
\iff x_g = -\frac{1}{2}\left((rb)^{-1}a_R + a_G\right) \pm \sqrt{\frac{1}{4}\left((rb)^{-1}a_R - a_G\right)^2 + (rb)^{-1}2^{2A}} & \\
\implies x_g = (rb)^{-1}\left(-\frac{1}{2}\left(a_R + rba_G\right) + \sqrt{\frac{1}{4}\left(a_R - rba_G\right)^2 + rb2^{2A}}\right) & \qquad (47)
\end{aligned}$$

where we in the last step used the fact that $x_g \geq 0$. From this it immediately follows that

$$\begin{aligned}
x_G &= b_G^{-1}x_g \\
&= b_R^{-1}r^{-1}\left(-\frac{1}{2}\left(a_R + rba_G\right) + \sqrt{\frac{1}{4}\left(a_R - rba_G\right)^2 + rb2^{2A}}\right), \qquad (48)
\end{aligned}$$

which in turn gives that

$$rx_G = b_R^{-1}\left(-\frac{1}{2}\left(a_R + rba_G\right) + \sqrt{\frac{1}{4}\left(a_R - rba_G\right)^2 + rb2^{2A}}\right). \tag{49}$$

We note that in the special case when $a_R = a_G = 0$ and $b_R = b_G = b_c$ we have that $x_G = (1/\sqrt{r})(2^A/b_c)$ and $rx_G = \sqrt{r}\,(2^A/b_c)$.

A.2 M as a function of A

The following equations

$$\begin{aligned} a_G + b_G x_G &= a_G + x_g \\ &= (rb)^{-1} \left(-\frac{1}{2}\left(a_R - rba_G\right) + \sqrt{\frac{1}{4}\left(a_R - rba_G\right)^2 + rb2^{2A}} \right) \end{aligned} \tag{50}$$

$$\begin{aligned} a_R + b_R x_R &= a_R + rbx_g \\ &= \frac{1}{2}\left(a_R - rba_G\right) + \sqrt{\frac{1}{4}\left(a_R - rba_G\right)^2 + rb2^{2A}} \end{aligned} \tag{51}$$

gives us the relation between M and A for a given fold change r

$$\begin{aligned} M = m_r(A) &= \log_2(rb) \\ &+ \log_2 \left(\frac{\frac{1}{2}\left(a_R - rba_G\right) + \sqrt{\frac{1}{4}\left(a_R - rba_G\right)^2 + rb2^{2A}}}{-\frac{1}{2}\left(a_R - rba_G\right) + \sqrt{\frac{1}{4}\left(a_R - rba_G\right)^2 + rb2^{2A}}} \right). \end{aligned} \tag{52}$$

E

Paper E

Calibration and assessment of channel-specific biases in microarray data with extended dynamical range

Henrik Bengtsson[1], Göran Jönsson[2], Johan Vallon-Christersson[2]

[1] Mathematical Statistics at the Centre for Mathematical Sciences, Lund University, Box 118, SE-221 00 Lund, Sweden. Email: hb@maths.lth.se. To whom correspondence should be made.
[2] Department of Oncology, Lund University, Barngatan 2, SE-221 85 Lund, Sweden.

Abstract

By scanning the same spotted oligonucleotide microarray at different photomultiplier tube (PMT) gains we have identified a channel-specific bias present in two-channel microarray data. The observed bias was very stable between subsequent scans of the same array although the PMT gain was greatly adjusted. This indicates that the bias does not originate from a step preceding the scanner detector parts. Between arrays the estimated bias varies slightly. When comparing bias estimates based on data from the same array, but from different scanners, we have found that different scanner models introduce different amounts of bias. So do various image analysis methods. We propose a scan protocol and a constrained affine model that allows us to identify and estimate the bias in each channel. Backward transformation removes the bias and brings the signals to the same scale. The result is that systematic effects such as intensity-dependent log-ratios are removed, but also that signal densities become much more similar. Average scan signals, which have a larger dynamical range and greater signal-to-noise ratio than the individual scans, can then be obtained. An implementation in R of the above calibration method is made available for free.

Keywords: microarray scanner; bias; dark noise; offset; zero level; affine model; multiscan calibration; normalization; dynamic range; photomultiplier tube; analog-to-digital converter; image analysis; intensity-dependent effects; background subtraction.

1 Introduction

The microarray technology provides a way of simultaneously measuring transcript abundances of $10^3 - 10^5$ genes from one or more individual cell or tissue samples. A microarray, also known as a gene chip, has well defined regions or features that each consists of immobilized sequences of DNA, which each is unique to a specific gene. These regions are referred to as *probes* [11]. When fluorophore labeled cDNA, referred to as *targets*, obtained by reverse transcription of mRNA extracted from the samples of interest, is let to hybridize to the probes for a few hours, each region on the microarray will specifically bind a certain amount of hybridized DNA unique to the corresponding gene. Depending on if a two-channel or single-channel microarray platform is used, either several and differentially labeled targets are hybridized to the same array, or different targets are each hybridized to an exclusive array using identical labels. Next, the array is scanned at different wavelengths to excite the fluorescent molecules using a light source, for instance a laser. Shortly after the fluorophores have been excited they emit photons, which are registered and quantified in each position by the scanner, which results in a high-resolution digitized image for each channel. Using image analysis methods, the pixels that belong to the regions that contain the probes are identified and averaged, and an estimate of the transcript abundance for each gene is obtained. Since these estimates are obtained from a complex measurement process of several steps, it is likely that the observed signals contain not only measurement noise, but also systematic variations of different kinds [5].

In this report we give evidence for a channel-specific bias that is introduced by the scanner and most likely its detector parts. In addition to this, our results indicate that the image analysis also contributes with a small bias. The effects channel-specific biases have on the downstream microarray analysis are many [8, 5]. Probably the most well known is the non-linear intensity-dependent bias in the log-ratios. We suggest a scan protocol and a model that will allow us to estimate the biases and calibrate the observed signals accordingly. The result will be that the intensity-dependent effects are removed, but also that the effective dynamical range of the scanner is increased several times.

The outline of this report is as follows. In Section 2, a model for the observed signals as a function of the probes/target concentration and the scanner settings is derived. In the same section, details on the two-channel microarray data set used are also given. Results are given in Section 3, followed by a discussion of possible reasons for the observed biases in Section 4. Conclusions are given in Section 5.

2 Materials and Methods

2.1 General model

Consider a microarray experiment involving genes $i = 1, 2, \ldots, I$ from RNA extracts $c = 1, 2, \ldots, C$. In single-channel microarrays each array measures the gene-expression levels in one RNA extract, whereas in two-color microarrays each array measures two RNA extracts, one in each channel. From now on, we will refer to each set of signals from each RNA extract as a *channel*. Let $\mu_{c,i}$ be the true gene-expression (transcription) level of gene i in channel c. Ideally, statistical analysis is done on these quantities. For instance, by comparing the relative abundances in two channels, that is $r_i = \mu_{1,i}/\mu_{2,i}$ for all genes i, it is possible to identify genes that are significantly differentially expressed ($r_i \neq 1$). However, in reality we do not observe the true expression levels, but only the quantified feature intensities $y_{c,i}$. The general relation between the observed and the true expression levels can be written as

$$y_{c,i} = f_c(\mu_{c,i}) + \varepsilon_{c,i}, \tag{1}$$

where $f_c(\cdot)$ is an unknown channel-specific function, which we refer to as the *measurement function*, that includes all steps in the microarray process. Moreover, we assume independent intensity-dependent error terms $\varepsilon_{c,i}$ such that $E[\varepsilon_{c,i}] = 0$. Because we want to do inference based on $\mu_{c,i}$, it must be possible to find the inverse of $f_c(\cdot)$, which is possible if it is strictly increasing.

To be able to find the form of $f_c(\cdot)$, high-quality calibration data from several stages along the microarray process is required. Here we will consider a simpler case. Split the overall measurement function into two parts. The first part, $x_{c,i} = g_c(\mu_{c,i})$, models the amount of light from spot i in channel c that enters the *photomultiplier tube* (PMT) [14] as a function of the transcription level of clone i in channel c. The second part, which is studied in this report, is $y_{c,i} = h_c(x_{c,i})$ and models the observed signal as a function of the amount of photons in channel c and spot i that enters the PMT. That is, it captures the nature of the scanner's light detector, but also the image analysis methods. We want to emphasize that the light from one spot does not necessarily originate solely from the fluorescent molecules that are attached to the hybridized target DNA. Light from other sources such as cross-hybridized target, intrinsic fluorescence from spot buffer, and scatter light may also contribute with photons of similar wavelengths.

Continuing, we will show that $h_c(\cdot)$ is almost perfectly affine. This measurement function depends also on the scanner settings, especially the scanner sensitivity, which is indicated below with the superscript (k). In other words,

$$y_{c,i}^{(k)} = a_c^{(k)} + b_c^{(k)} x_{c,i} + \varepsilon_{c,i}^{(k)},\ \forall i, c \tag{2}$$

where for each fixed scanner setting k, $a_c^{(k)}$ and $b_c^{(k)}$ are channel-specific bias and scale parameters, respectively. Assume that $x_{c,i}$ is *fix* for all PMT voltages.

Note that the above relationship is not necessarily linear ($a_c = 0$), which has important implications on downstream analysis. For instance, when spotted as well as in-situ synthesized microarray are used, it is common to do statistical analysis on the log-ratios $M_i = \log_2(y_{R,i}/y_{G,i})$ and the log-intensities $A_i = \frac{1}{2}\log_2(y_{R,i}y_{G,i})$ for all genes i [10], where we for convenience have denoted the two channels to be compared by R and G, which are mnemonics for the red and the green dyes commonly used in two-channel microarray data, although such comparisons are not limited to within-array measurements. One of the rationales for this bijective transform is that under ideal conditions the main measure of interest, the fold change, is contained in one variable only, i.e. in M_i. However, even if we could identify $g_c(\cdot)$ and calibrate data for that part accordingly, a channel specific bias introduced by $h_c(\cdot)$ will introduce so called intensity-dependent dye bias in the observed log-ratios. Commonly observed intensity-dependent effects in the log-ratios [23] can be explained by the fact that the logarithm is taken on affinely transformed signals [12, 8, 5].

2.2 Data

2.2.1 Arrays and hybridization

The results presented here are based on data from nine different hybridizations performed using spotted oligonucleotide microarrays, referred to here by the letters A to H. Array A and B were hybridized in October 2003. Arrays C-G were hybridized the following day and array H was hybridized seven weeks later. All arrays contain the same human oligonucleotide set (QIAGEN) and all have an identical layout of 12-by-4 print-tip groups each containing 34-by-34 (1156) spots. In total there are 55488 spots on each array. The average (GenePix) spot area is 45-50 pixels and the average center-to-center distance between the spots is approximately 12-13 pixels (120-130 μm). Arrays were produced by the SWEGENE DNA Microarray Resource Centre, Department of Oncology at Lund University using a MicroGrid II 600R arrayer fitted with MicroSpot 10K pins (BioRobotics). All arrays, except array H, were spotted in the same print batch on UltraGAPS^{TM}coated slides (Corning Incorporated) during August 2003. Array H was spotted in October the same year. Printing was performed in a temperature (18-20°C) and humidity (44-49% RH) controlled area. After printing was completed, arrays were left in a desiccator to dry for 48 hours, rehydrated for 1 second over steaming water, snap dried on a hot plate (98°C), UV-cross-linked (800 mJ/cm^2) and subsequently hybridized with various test and reference RNA samples. Samples were labeled, purified and hybridized using Pronto!^{TM} Plus System 6 (Corning Incorporated) according to manufacturer's instructions.

2.2.2 Scanning

Each array was scanned at four different PMT settings on two different types of scanners. First the arrays were scanned on an Agilent G2505A DNA microarray scanner (Agilent Technologies) at PMT gains 100%, 30%, 50%, and 80% (in that order). The so called *dark offset*, intentionally added by the Agilent scanner to all signals [1, p. 18], has been uninstalled and is therefore not the reason for the findings in this report. Arrays were then re-scanned on an Axon GenePix 4000A scanner (Axon Instruments) at PMT gains 600, 700, 800, and 500 volts (in that order), except for array A, which was scanned at 700, 800, 500 and 600 volts, and array H, which was scanned at 600, 400, 500 and 700 volts. Thus, the images obtained by the Axon scanner were bleached more than the preceding ones obtained by the Agilent scanner. For both scanners, the power of the 532 nm and the 635 nm lasers was set to 100% and the scan resolution to 10 μm/pixel. Moreover, a one-pass (both channels scanned simultaneously) and one-sample-per-pixel (Lines To Average or LTA equals one) procedure was used for all scans. The Agilent scanner has a special loading mechanism for microarrays, which allows automatic scanning of subsequent arrays without human intervention. However, due to limitations in the software or the scanner, each such batch of arrays can only be scanned at a single PMT gain. To scan at more PMT gains with the Agilent scanner, it was therefore necessary to eject and reload the arrays between different PMT settings, which means that the alignment between the scanned images may not be perfect. Contrary, for the Axon scanner the arrays were put in the scanner one by one, then scanned at all PMT settings without being moved.

2.2.3 Image analysis - spot segmentation and registration

To quantify the foreground and the background signals, the scanned images (65536 gray scales and approximately 2000-by-5600 pixels) were analyzed using both the Axon GenePix Pro v4.1.1 software (Axon Instruments) and the Spot v2 software [22, 2]. We first analyzed each image with GenePix. For each of them, the grid and spot positions were manually set and then the alignment was optimized by GenePix. These positions were re-entered and re-optimized in Spot with visual inspection to verify the correctness. Moreover, for each individual scan the image analysis software was let to find the optimal spot segmentation. Thus, what is defined as a foreground pixel may vary with PMT setting although the images are from the same array. We decided on this schema for various reasons. The first reason was that the Agilent arrays are loaded and unloaded between subsequent scans and therefore require separate spot segmentations. To be able to compare the results from the Axon and the Agilent scanner, we choose the same procedure for the images scanned on the Axon scanner, even though, the optimized segmentation for the strongest image could have been reused. We further believe that this allows us compare Spot and GenePix more fairly. For both Spot and GenePix the median spot pixel intensity was used as for the foreground signal. Background estimates were not considered in this analysis. No spot signals were discarded in analysis.

2.2.4 Data analysis

All further analysis was carried out using R [20, 13] and the aroma package (formerly known as the com.braju.sma package) [4] in which the methods presented here have been included.

2.3 Constrained model

Model (2) is not identifiable. Consider the case where the same array has been scanned at K different PMT settings. Let $\mathbf{y}_{c,i} = (y_{c,i}^{(1)}, y_{c,i}^{(2)}, \ldots, y_{c,i}^{(K)})$ be the vector of the K quantified signals for gene i and channel c. In the noise-free case it follows from (2) that $\mathbf{y}_c = \{\mathbf{y}_{c,i}\}_{i=1}^{I}$ lies on the line $L(\mathbf{a}_c, \mathbf{b}_c)$ in $\mathbb{R}^K$, which has direction $\mathbf{b}_c = (b_c^{(1)}, b_c^{(2)}, \ldots, b_c^{(K)})$ and goes through the point $\mathbf{a}_c = (a_c^{(1)}, a_c^{(2)}, \ldots, a_c^{(K)})$. The $2K$ parameters of $\mathbf{a}_c$ and $\mathbf{b}_c$ are not identifiable, since L has only $2K-2$ degrees of freedom. In fact, any transformation $\mathbf{b}_c \leftarrow k \cdot \mathbf{b}_c$, and $\mathbf{a}_c \leftarrow \mathbf{a}_c + l \cdot \mathbf{b}_c$, where k and l are scalars, will leave L intact. In this paper, we make $\mathbf{a}_c$ and $\mathbf{b}_c$ identifiable by choosing k and l so that $b_c^{(1)} = 1$ and $\mathbf{a}_c$ is the point on L closest (in Euclidean norm) to the (diagonal) line $L' = \{e_c(1, \ldots, 1); e \in \mathbb{R}\}$. The choice of $\mathbf{a}_c$ can be motivated by looking at the observed data. By a first inspection we observe that the bias $a_c^{(k)}$ in model (2) is not varying much when the PMT gain is changed. To demonstrate this, $(y_R^{(k)}, y_R^{(l)})$ have been plotted in a unique color together with the correspond-

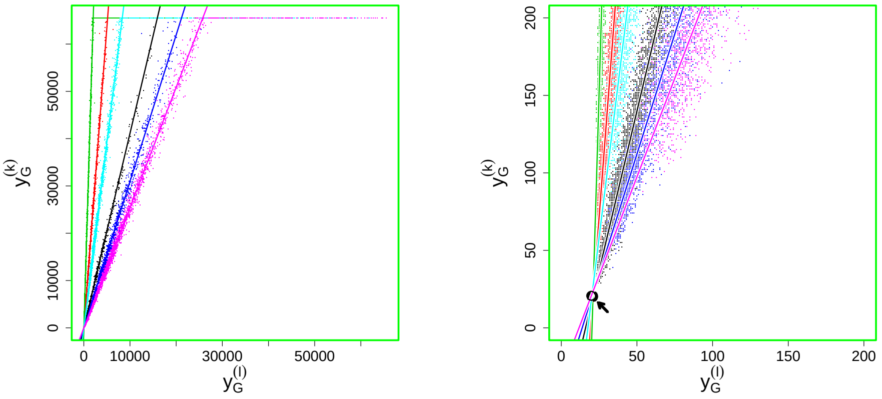


Figure 1: Scanning the arrays at different PMT gains indicates that there is an affine relationship between quantified fluorescent intensities and concentration of fluorophores. *Left:* Observed signals in the red channel for PMT pairs $(800, 500)$, $(700, 500)$, $(800, 600)$, $(600, 500)$, $(700, 600)$, and $(800, 700)$ are shown in green, red, cyan, black, blue and magenta, respectively. An affine model is estimated for each pair and displayed as a line. Data points at the very top are saturated and ignored. *Right:* A zoom-in of the same data. For *each* pair the estimated $(\hat{a}_G^{(l)}, \hat{a}_G^{(k)})$, which corresponds to the true origin, has been highlighted with a circle. All lines seem to intersect at the same point on the $y_c^{(k)} = y_c^{(l)}$ line. Shown are signals from the green channel on array A quantified by GenePix from Axon scanner images.

ing fitted line for each of the six possible PMT pairs in Figure 1. The scatter plot to the left shows all observations and the one to the right zooms in on the low-intensity spots. First, the close fit of data to the lines is evidence that the scanner is linear in its dynamical range. Second, all lines go through approximately the same point, lets call it (e_c, e_c), which suggests that there is a common PMT-independent bias e_c to all scans. More precisely, split the bias term into two parts, one dependent and one independent on the PMT gain according to

$$a_c^{(k)} = d_c^{(k)} + e_c;\ k = 1, 2, \dots, K, \tag{3}$$

and define $\mathbf{d}_c = (d_c^{(1)}, d_c^{(2)}, \dots, d_c^{(K)}) \in \mathbb{R}^K$. Then, data indicates that $\|\mathbf{d}_c\| \approx 0$, where $\|\cdot\|$ is the norm in, say, L_2 (Euclidean distance). Let $\mathbf{d} = \mathbf{y} - e(1, \dots, 1)$ where $\mathbf{y} \in L$ and $e \in \mathbb{R}$. The constraint that $\mathbf{a}_c$ is the point of L closest to L' can be formulated as

$$\mathbf{d}_c = \arg\min_{\mathbf{d}} \|\mathbf{d}\| \tag{4}$$

where minimization is with respect to $\mathbf{y}$ and e. Equivalently, this means that $\mathbf{d}_c$ is orthogonal to $\mathbf{b}_c$ and $(1, \dots, 1)$. The above can be interpreted geometrically as follows. By definition, $\mathbf{a}_c$ is a point on the line $L(\mathbf{a}_c, \mathbf{b}_c)$. Similarly, $\mathbf{e}_c = e_c(1, \dots, 1)$ is a point on the diagonal line that goes through $(0, \dots, 0)$ and $(1, \dots, 1)$ in $\mathbb{R}^K$, i.e. L'. Now, minimizing $\mathbf{d}_c$ according to (4) is the same as finding the shortest distance between the line L and the diagonal line, which is also the distance between the two points $\mathbf{a}_c$ and $\mathbf{e}_c$. From the geometric interpretation it is also clear that, in order for the parameters to be uniquely identifiable, the line L *must not* be parallel to the diagonal line, that is, $b_c^{(k)}$ must be different from $b_c^{(1)}$ for some k. A robust estimate of L was proposed in [5], using *iteratively reweighted principal component analysis* (IWPCA). This estimate of L, together with the above parametrization of $\mathbf{a}_c$ and $\mathbf{b}_c$, give us estimates $\hat{\mathbf{a}}_c$ and $\hat{\mathbf{b}}_c$ of all $2K-2$ parameters of $\mathbf{a}_c$ and $\mathbf{b}_c$, as well as an estimate of $\hat{e}_c$ of e_c.

Let us illustrate the parametrization and estimation procedure for $K = 2$. Since two (non-parallel) lines will always intersect, the constraint (4) degenerates to the assumption that $\mathbf{d}_c = \mathbf{0}$, or equivalently, that $a_c^{(1)} = a_c^{(2)} = e_c$. In the noise-free case the line L is described by

$$y_{c,i}^{(2)} = \alpha_c + \beta_c y_{c,i}^{(1)};\ i = 1, 2, \dots, I, \tag{5}$$

where $\alpha_c = e_c(1 - b_c^{(2)}/b_c^{(1)})$ and $\beta_c = b_c^{(2)}/b_c^{(1)}$. By setting $y_{c,i}^{(2)} = y_{c,i}^{(1)}$ in (5) and applying the constraint $b_c^{(1)} = 1$, we get that $\mathbf{a}_c = (e_c, e_c)$ and $\mathbf{b}_c = (1, \beta_c)$ where $e_c = \alpha_c/(1 - \beta_c)$. To further illustrate the stability of the PMT independence, the parameters (e_c, β_c) have been estimated for each of the six PMT pairs *independently* based on data from array A scanned by the Axon scanner and quantified by GenePix. The various estimates for both channels are listed in Table 1. The average estimate of the bias across all PMT pairs in the red channel was $\bar{e}_R = 18.0$ (with standard deviation 1.12). For the green channel the average bias estimate was $\bar{e}_G = 20.3$ (with standard deviation 0.80). The small standard deviations confirm that $\mathbf{d}_c$ is indeed small.

PMT pair	$\hat{e}_R$	$\hat{\beta}_R$	$\hat{e}_G$	$\hat{\beta}_G$
(800,500)	18.9	32.4	20.7	32.5
(700,500)	19.0	13.0	20.9	12.8
(800,600)	17.3	7.1	19.8	7.9
(600,500)	19.0	4.5	20.8	4.1
(700,600)	17.6	2.9	20.9	3.1
(800,700)	16.3	2.5	18.9	2.6
mean	18.0 ± 1.12		20.3 ± 0.80	

Table 1: Red and green channel estimates of the bias and the slope for each PMT voltage pair together with the mean and the standard deviation of all estimates. The small standard deviation is evidence that the bias to a large extent is independent of the PMT settings. Data shown originates from array A scanned on Axon and analyzed with GenePix.

2.4 Calibration

Given estimates of $a_c^{(k)}$ and $b_c^{(k)}$, data can be calibrated using backward transformation. Let

$$\tilde{y}_{c,i}^{(k)} = \frac{y_{c,i}^{(k)} - a_c^{(k)}}{b_c^{(k)}}, \tag{6}$$

$$\tilde{\varepsilon}_{c,i}^{(k)} = \frac{\varepsilon_{c,i}^{(k)}}{b_c^{(k)}} \tag{7}$$

be the backward transformed observed signal and the rescaled error terms, respectively, for all c, i and k. The affine model (2) can then be rewritten as

$$\tilde{y}_{c,i}^{(k)} = x_{c,i} - \tilde{\varepsilon}_{c,i}^{(k)};\ \forall c, i, k. \tag{8}$$

Moreover, let

$$\bar{\tilde{y}}_{c,i} = \frac{1}{K} \sum_{k=1}^{K} \tilde{y}_{c,i}^{(k)} \tag{9}$$

be the average backward transformed signal for gene i in channel c. Now, if $E[\varepsilon_{c,i}^{(k)}] = 0$, then $E[\tilde{\varepsilon}_{c,i}^{(k)}] = 0$ and hence

$$E[\bar{\tilde{y}}_{c,i}] = x_{c,i} \tag{10}$$

when all $a_c^{(k)}$ and $b_c^{(k)}$ are known. Thus, if (6) is applied with estimates of $a_c^{(k)}$ and $b_c^{(k)}$ that are consistent as $I \to \infty$, and the error terms have zero mean, the mean of the backward transformed signals will converge to $x_{c,i}$ as I grows. Even though $E[\tilde{y}_{c,i}]$ is not observable, we can estimate it consistently by increasing the number of scans K. Inspection of the residuals of calibrated signals (not shown) indicates that the variance of the calibrated noise is independent of PMT setting, that is $V[\tilde{\varepsilon}_{c,i}^{(k)}] = \sigma_{c,i}^2$. Assuming independent noise terms, the variance of the sample mean is

$$V[\bar{\tilde{y}}_{c,i}] = \frac{1}{K} \sigma_{c,i}^2, \tag{11}$$

or equivalent, the standard deviation of the calibrated mean spot signal decrease as $1/\sqrt{K}$ where K is the number of scans. In summary, we obtain consistent estimates (up to a multiplicative constant) of all $x_{c,i}$ with increasing I and K.

Finally, signals that are saturated by the scanner have to be excluded before calculating the average. If the quantified signal for a spot happens to be saturated in all scans, then that spot is marked as saturated, which still may be informative when compared to other non-saturated signals.

3 Results

3.1 Parameter estimates

For every possible combination of array, scanner and signal quantification method (image analysis or raw pixel intensities), we estimated the parameters $\mathbf{a}_c$ (including e_c and $\mathbf{d}_c$) and $\mathbf{b}_c$ in model (2)-(4) for both channels according to the algorithm described in Section 2.3. To better understand the properties of the estimates, we use a bootstrap approach to obtain not only bias corrected estimates, but also their standard deviations. For GenePix and Spot quantified signals a bootstrap sample of size 100 was used. For the estimates based on the raw pixel intensities a different approach was taken. Because the number of pixels for one scan is about 10^7 (per channel) and we had four scans our computer system limited us to estimate the model based on a subset of 10^6 pixel intensities. This was done for 100 random subsets and the mean and standard deviation of the parameter estimates were calculated, much like the above bootstrap method.

The mean and the standard deviation (within parentheses) of $\hat{e}_c$ and $||\hat{\mathbf{d}}_c||$ for all possible setups are listed in Table 2 and Table 3. The mean and standard deviation of $\hat{e}_c$, and the corresponding median and median absolute deviation (MAD), across all arrays are shown in Table 4 and Table 5, respectively.

3.1.1 Comparison of arrays

The bias estimates for all bootstrap replicates in Table 2 and Table 3 have been depicted as box plots in Figure 2. Considered that the signals are in range is of 0 to 65535, the bias estimates are very stable between different arrays. The bias estimates span 9.8 units (0.15‰) in the red channel and 7.8 units (0.12‰) in the green channel. Next we will study the differences in bias estimates between scanners, image analysis methods, and channels.

3.1.2 Comparison of scanners

For the two scanners, we found that the estimated biases based on signals obtained by the Agilent scanner are consequently higher than the estimates from the Axon scanner. The box plots of their differences in the common bias e_c (for each bootstrap sample) between the Agilent and the Axon scanner in Figure 3 confirm this. See also Table 4 and Table 5. The significant difference could be an effect of scan

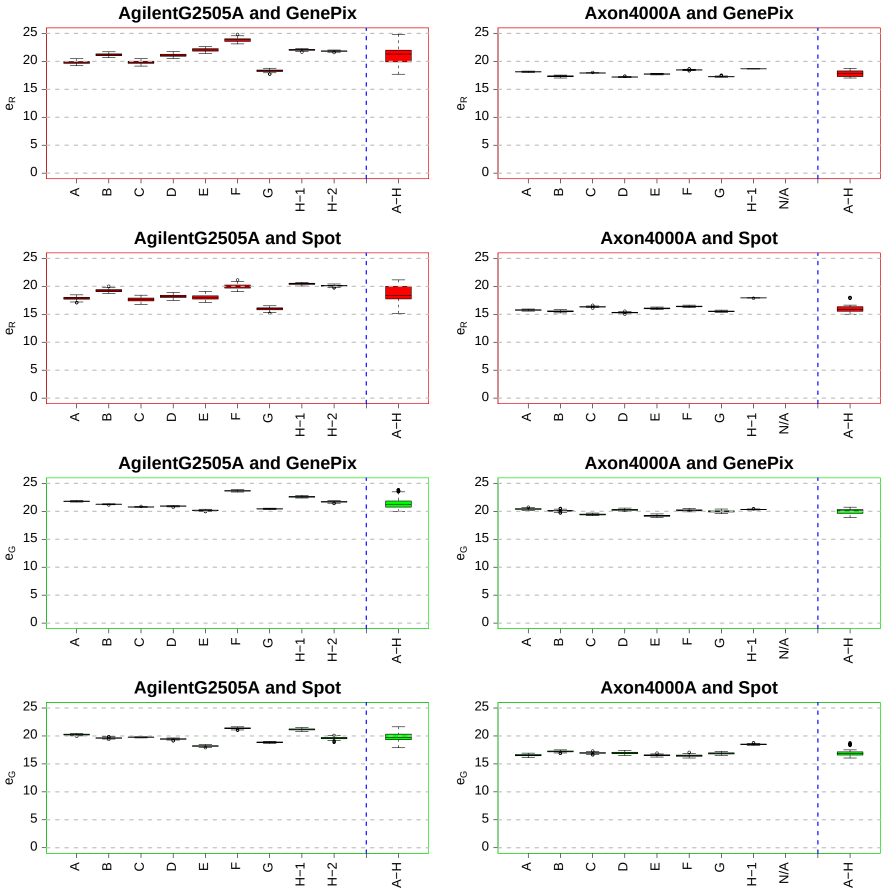


Figure 2: Estimated biases e_c for each array and for all arrays together from the Agilent scanner (left column) and the Axon scanner (right column). The top four graphs are for the red channel and the bottom four are for the green channel. For each channel the top two are estimates based on signals quantified by GenePix and the bottom two on signals quantified by Spot. See also Table 2 and Table 3.

array	image	Agilent		Axon	
	method	$\hat{e}_R$	$\|\hat{\mathbf{d}}_R\|$	$\hat{e}_R$	$\|\hat{\mathbf{d}}_R\|$
A	Spot	17.9 ± 0.289	1.14 ± 0.148	15.8 ± 0.090	4.40 ± 0.145
A	GenePix	19.8 ± 0.253	0.82 ± 0.162	18.1 ± 0.058	1.27 ± 0.095
A	pixels	44.9 ± 0.036	22.85 ± 0.043	19.0 ± 0.032	0.68 ± 0.052
B	Spot	19.2 ± 0.255	1.05 ± 0.165	15.5 ± 0.121	4.01 ± 0.255
B	GenePix	21.2 ± 0.225	1.83 ± 0.203	17.3 ± 0.114	1.72 ± 0.253
B	pixels	42.8 ± 0.049	20.71 ± 0.051	17.8 ± 0.034	2.06 ± 0.079
C	Spot	17.6 ± 0.366	2.00 ± 0.192	16.3 ± 0.076	4.83 ± 0.191
C	GenePix	19.8 ± 0.284	0.94 ± 0.209	17.9 ± 0.034	2.11 ± 0.135
C	pixels	21.0 ± 0.138	1.48 ± 0.079	18.8 ± 0.022	1.24 ± 0.061
D	Spot	18.2 ± 0.301	1.56 ± 0.103	15.3 ± 0.093	4.10 ± 0.149
D	GenePix	21.1 ± 0.274	0.95 ± 0.139	17.2 ± 0.049	1.74 ± 0.090
D	pixels	25.0 ± 0.522	4.10 ± 0.358	18.3 ± 0.025	1.02 ± 0.044
E	Spot	18.0 ± 0.428	1.15 ± 0.109	16.1 ± 0.101	3.01 ± 0.131
E	GenePix	22.1 ± 0.268	1.77 ± 0.223	17.7 ± 0.064	1.02 ± 0.096
E	pixels	24.7 ± 0.144	5.51 ± 0.169	18.7 ± 0.024	0.39 ± 0.025
F	Spot	19.9 ± 0.423	0.48 ± 0.163	16.4 ± 0.087	3.76 ± 0.138
F	GenePix	23.8 ± 0.316	1.47 ± 0.258	18.5 ± 0.060	1.07 ± 0.106
F	pixels	24.2 ± 0.131	1.37 ± 0.101	19.3 ± 0.026	0.43 ± 0.038
G	Spot	16.0 ± 0.300	2.30 ± 0.166	15.5 ± 0.077	3.54 ± 0.114
G	GenePix	18.3 ± 0.208	1.88 ± 0.198	17.3 ± 0.070	1.57 ± 0.134
G	pixels	19.1 ± 0.096	1.48 ± 0.080	18.3 ± 0.026	0.66 ± 0.038
H-1	Spot	20.4 ± 0.161	2.12 ± 0.073	17.9 ± 0.025	1.35 ± 0.043
H-1	GenePix	22.1 ± 0.124	2.41 ± 0.097	18.7 ± 0.024	0.81 ± 0.041
H-1	pixels	44.9 ± 0.513	17.68 ± 0.528	19.1 ± 0.013	0.78 ± 0.027
H-2	Spot	20.1 ± 0.141	0.33 ± 0.040	N/A	N/A
H-2	GenePix	21.8 ± 0.087	0.22 ± 0.084	N/A	N/A
H-2	pixels	21.8 ± 0.047	0.26 ± 0.042	N/A	N/A

Table 2: Bootstrapped parameter estimates for the red channel with standard deviations for each combination of array, image method (or pixel intensities), and scanner.

order, that is, all arrays were first scanned on the Agilent scanner and then the Axon scanner. The arrays in hand were part of a much bigger project based solely on Agilent scanned data. To keep a consistent scan protocol and to avoid the risk of getting bleached signals, we could not simply balance the experimental design by letting some arrays be scanned in the reverse order. Instead, to test for scan order trends, we scanned one array (H) first on Agilent, then on Axon and then again on Agilent. The two Agilent-Axon comparisons are labeled H-1 and H-2'. No apparent trend was found.

array	image	Agilent		Axon	
	method	$\hat{e}_G$	$\|\hat{\mathbf{d}}_G\|$	$\hat{e}_G$	$\|\hat{\mathbf{d}}_G\|$
A	Spot	20.2 ± 0.106	0.51 ± 0.058	16.6 ± 0.165	6.31 ± 0.302
A	GenePix	21.8 ± 0.068	0.20 ± 0.045	20.4 ± 0.123	0.88 ± 0.242
A	pixels	33.6 ± 0.050	10.52 ± 0.048	22.8 ± 0.048	1.57 ± 0.093
B	Spot	19.6 ± 0.094	0.90 ± 0.055	17.2 ± 0.127	2.20 ± 0.231
B	GenePix	21.3 ± 0.046	0.94 ± 0.086	20.1 ± 0.152	1.20 ± 0.326
B	pixels	35.7 ± 0.153	12.85 ± 0.146	21.9 ± 0.041	2.04 ± 0.087
C	Spot	19.8 ± 0.050	1.21 ± 0.073	17.0 ± 0.127	2.75 ± 0.254
C	GenePix	20.8 ± 0.039	0.99 ± 0.086	19.4 ± 0.100	1.11 ± 0.218
C	pixels	21.3 ± 0.024	0.78 ± 0.035	22.0 ± 0.040	2.39 ± 0.082
D	Spot	19.4 ± 0.097	1.49 ± 0.073	17.0 ± 0.194	1.24 ± 0.376
D	GenePix	20.9 ± 0.059	0.80 ± 0.095	20.3 ± 0.126	2.31 ± 0.286
D	pixels	21.6 ± 0.105	0.21 ± 0.124	22.1 ± 0.046	3.10 ± 0.092
E	Spot	18.2 ± 0.121	3.29 ± 0.129	16.5 ± 0.132	2.88 ± 0.240
E	GenePix	20.2 ± 0.081	1.92 ± 0.128	19.2 ± 0.137	0.74 ± 0.264
E	pixels	21.2 ± 0.017	0.13 ± 0.046	21.0 ± 0.032	0.94 ± 0.058
F	Spot	21.4 ± 0.125	0.58 ± 0.103	16.5 ± 0.181	4.79 ± 0.327
F	GenePix	23.7 ± 0.084	1.32 ± 0.116	20.2 ± 0.132	0.34 ± 0.199
F	pixels	24.1 ± 0.019	1.57 ± 0.026	22.4 ± 0.039	1.87 ± 0.071
G	Spot	18.9 ± 0.088	1.62 ± 0.088	16.9 ± 0.149	3.72 ± 0.258
G	GenePix	20.4 ± 0.049	1.10 ± 0.088	20.0 ± 0.164	0.84 ± 0.262
G	pixels	21.0 ± 0.017	0.75 ± 0.024	22.2 ± 0.045	2.60 ± 0.085
H-1	Spot	21.2 ± 0.134	3.25 ± 0.080	18.5 ± 0.090	2.00 ± 0.170
H-1	GenePix	22.6 ± 0.100	3.47 ± 0.107	20.3 ± 0.043	0.63 ± 0.100
H-1	pixels	42.8 ± 1.417	16.68 ± 1.843	21.3 ± 0.033	1.82 ± 0.070
H-2	Spot	19.6 ± 0.229	0.16 ± 0.077	N/A	N/A
H-2	GenePix	21.7 ± 0.104	0.55 ± 0.076	N/A	N/A
H-2	pixels	21.5 ± 0.086	0.29 ± 0.021	N/A	N/A

Table 3: Bootstrapped parameter estimates for the green channel with standard deviations for each combination of array, image method (or pixel intensities) and scanner.

3.1.3 Comparison of image analysis methods

Furthermore, estimates of the common bias e_c based on GenePix quantified signals are consistently greater than the corresponding ones based on Spot signals, cf. Tables 2 and 3. See also Tables 4 and 5. The box plots in Figure 4 show differences in estimates of the common bias (for each bootstrap sample) between GenePix and Spot. The difference may be explained by the fact that the two applications use different spot segmentation algorithms [22, 3]. Because the con-

image	$\hat{e}_R$		$\hat{e}_G$	
method	Agilent	Axon	Agilent	Axon
Spot	18.6 ± 1.40	16.1 ± 0.792	19.8 ± 0.958	17.0 ± 0.633
GenePix	21.1 ± 1.55	17.8 ± 0.529	21.5 ± 1.05	20.0 ± 0.433
pixels	29.8 ± 10.4	18.7 ± 0.446	27.0 ± 7.76	22.0 ± 0.537

Table 4: Mean and standard deviation of the bias estimates of all arrays and for each signal quantification method.

image	$\hat{e}_R$		$\hat{e}_G$	
method	Agilent	Axon	Agilent	Axon
Spot	18.4 ± 1.49	15.9 ± 0.619	19.7 ± 0.854	16.9 ± 0.421
GenePix	21.3 ± 1.27	17.9 ± 0.823	21.3 ± 0.822	20.1 ± 0.315
pixels	24.6 ± 5.38	18.8 ± 0.527	21.6 ± 0.817	22.1 ± 0.386

Table 5: Median and MAD (median absolute deviation) of the bias estimates of all arrays and for each signal quantification method.

centration of fluorophores is not homogeneous across a spot, the result is that the distribution of pixel intensities will vary with segmentation method. This effect can be more profound for spots with strong donut effects. Robust estimates such as the median pixel value will to some extend protect against this, but not completely. It has been suggested [7] that the *median of* (pixel) *ratios* is a better estimate of the ratio of hybridized cDNA than the *ratio of median* (pixels). However, the former requires that the images are perfectly aligned with respect to shift, rotation, shear and so on. Also, it applies exclusively to two sample comparisons. Because of this, we do not believe that pixel-ratio signals are useful in practice.

3.1.4 Pixel-based estimates

To better understand the underlying reasons for the observed channel biases, the proposed affine model was also applied to pixel intensities (instead of spot signals). The estimated biases for the red and the green channel for different arrays using multidimensional IWPCA based on *pixel values* are shown in Tables 2 and 3. Except for the green channel in the second scan round of array H, the pixel-based estimates are consistently higher than the estimates based on GenePix and the Spot foreground signals. As noted above, pixel-based estimates are very sensitive to image distortions. This is especially a concern for the Agilent scanner since it

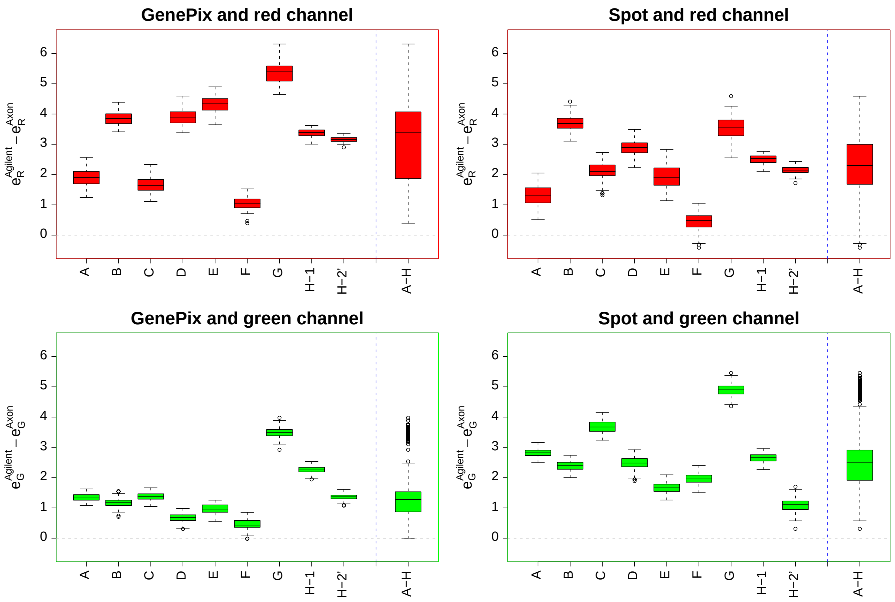


Figure 3: Differences in the common biases e_c between the Agilent and the Axon scanners for each separate array and for all arrays together. For both the red (top) and the green channel (bottom), the biases estimated from GenePix quantified signals (left) are significantly greater than the biases estimated from Spot quantified signals (right). The exception is a small fraction of the bootstrap estimates from array F.

reloads the arrays between subsequent scans. To check this, we did a test where a test person with experience in microarray analysis was asked to subjectively rank how badly aligned the four images in the red channel with different PMT gains from the Agilent scanner were for each of the (unlabeled) nine arrays. The person rated the images from arrays A, B, D, and H-1 to be “extremely” misaligned. The images from array E were considered to be “quite” misaligned, and the images from array C to be “slightly” misaligned. For the rest of the arrays the images were considered to be aligned (less than one pixel off). This blind-folded test is perfectly in line with the discrepancies between pixel estimates and the others listed in Tables 2, which confirms our hypothesis. Another disadvantages of estimating the biases based on pixel intensities is that it is extremely memory and time consuming. For instance, estimating the parameters based on 10^6 pixels

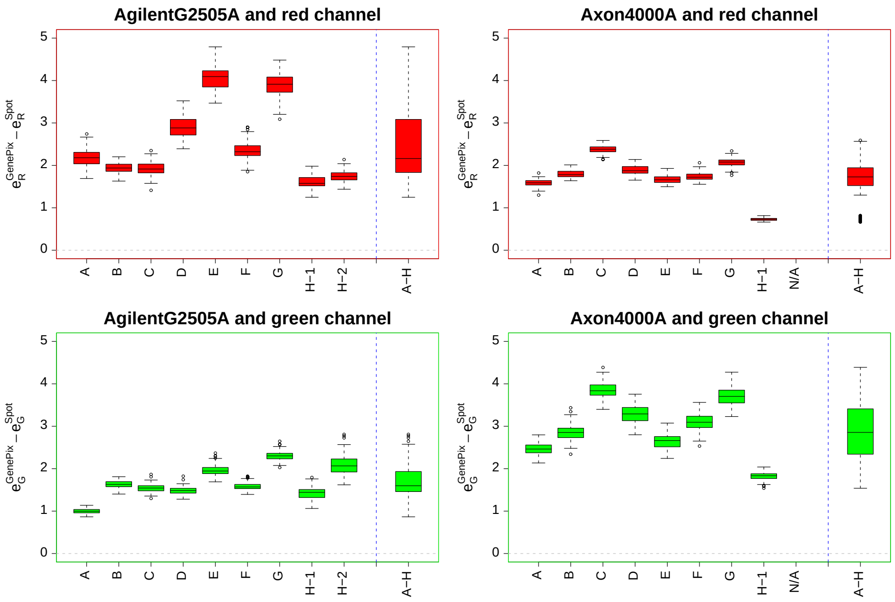


Figure 4: Differences in the common biases e_c between the GenePix and the Spot image analysis method for each separate array and for all arrays together. For both the red (top) and the green channel (bottom), the biases estimated from Agilent signals (left) are significantly greater than the biases estimated from Axon signals (right).

took approximately 50 times longer than to estimate them based on 55488 quantified signals, which is because a greater number of data points are fitted, but also because the pixel intensities had to be reloaded and resampled before each estimation round due to memory constraints.

3.1.5 Comparison of channels

As Figure 5 shows, the common bias e_c is greater in the green channel than the red channel, especially for GenePix quantified signals, when estimated based on data from the Axon scanner. For the Agilent scanner this trend is less clear, though the Spot quantified signals clearly seem to give higher bias in the green that the red channel. Furthermore, the biases in the red and the green channels seem to be stable between arrays, which give further evidence to our hypothesis that the bias originates from the scanner (and/or the image analysis methods).

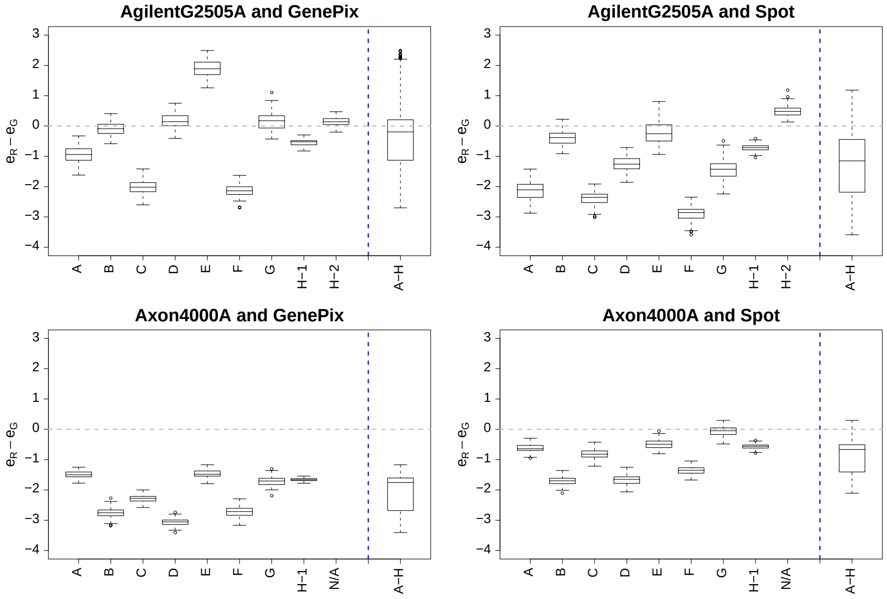


Figure 5: Differences in the common biases e_c between the red and the green channel for each separate array and for all arrays together. At the top is data from the Agilent scanner and at the bottom from the Axon scanner. Data in the left column by GenePix and data in the right column was quantified by Spot. For data from the Axon scanner the common bias is greater in the green channel than the red channel, especially for GenePix quantified signals. For the Agilent scanner this trend is less clear although the Spot quantified signals clearly seem to have a higher bias in the green than the red channel.

3.1.6 Deviation in bias estimates between PMT gains

In Figure 6 the distribution of the "bias residuals" $d_c^{(k)}$ are depicted for different scans k and channels c, for each separate array, but also all arrays together, and for both scanners and both image analysis methods. Each array is presented in increasing PMT order. The most apparent effect is that the estimates based on signals from the Axon scanner and especially those quantified by the Spot software are greater than for the others, cf. Table 2 and Table 3. Why this is so we do not know. For some arrays the estimates from the red and the green channels are strongly correlated, but it is not clear to us when this occurs. Although not in general, for some combinations of scanner and image analysis method, there is a trend in the PMT order (or possibly scan order). Again, we do not know why. To summarize, we have by means of exploratory data analysis (not shown) tried to understand what sometimes looks like patterns in the $d_c^{(k)}$:s, but we found no apparent relationships. However, systematic effects might indicate that $d_c^{(k)}$ could be modeled further.

3.2 Calibration

When data was calibrated according to the backward transformation in (6)-(9), estimates (up to a scale factor) of all $x_{c,i}$:s were obtained. Since we do not know the true values, we cannot verify the estimates directly. However, to some extent, we may do it indirectly by looking for remaining systematic effects in the log-ratios, but also by comparing the empirical densities of the calibrated scans. For a detailed study on various systematic effects an affine transformation introduces in the log-ratios, see [5]. For instance, here the amount of intensity-dependent curvature in the log-ratios is related to the bias and the relative scale factor via the product $e_c(1 - b_c^{(k)}/b_c^{(l)})$ assuming $\|\mathbf{d}_c\| = 0$. To demonstrate this, we have for various PMT pairs compared the within-channel log-ratios and log-intensities

$$M_{c,i}^{(k,l)} = \log_2(y_{c,i}^{(k)}/y_{c,i}^{(l)}) \tag{12}$$

$$A_{c,i}^{(k,l)} = \frac{1}{2}\log_2(y_{c,i}^{(k)}y_{c,i}^{(l)}), \tag{13}$$

respectively, with the corresponding ones for the backward transformed data, which we denote by $\tilde{M}_{c,i}^{(k,l)}$ and $\tilde{A}_{c,i}^{(k,l)}$. The log-ratios versus the log-intensities for the *raw signals* of all six PMT pairs are shown in the left scatter plot in Figure 7. The corresponding plot for the *backward transformed signals* is shown to the very right. For each of the six data clouds, the curvature, but also the overall bias, in the log-ratios is removed. To further underline the effect that a channel-specific *bias* has, we have calculated the log-ratios for the *bias-subtracted* signals (no rescaling), which makes model (2) become *linear*. As seen in the middle scatter plot, the curvature introduced by the bias and the logarithm is removed. The overall bias in the log-ratios that remains is $\log_2(b_c^{(k)}/b_c^{(1)})$ and is removed when the signals are rescaled. Note that it is not correct to shift only the log-ratios toward zero, because then the log-intensities will be incorrect. The various M versus A scatter plots become very similar and so do the four empirical density functions of the signals, as seen in Figure 8. The small bumps at high intensities are due to the

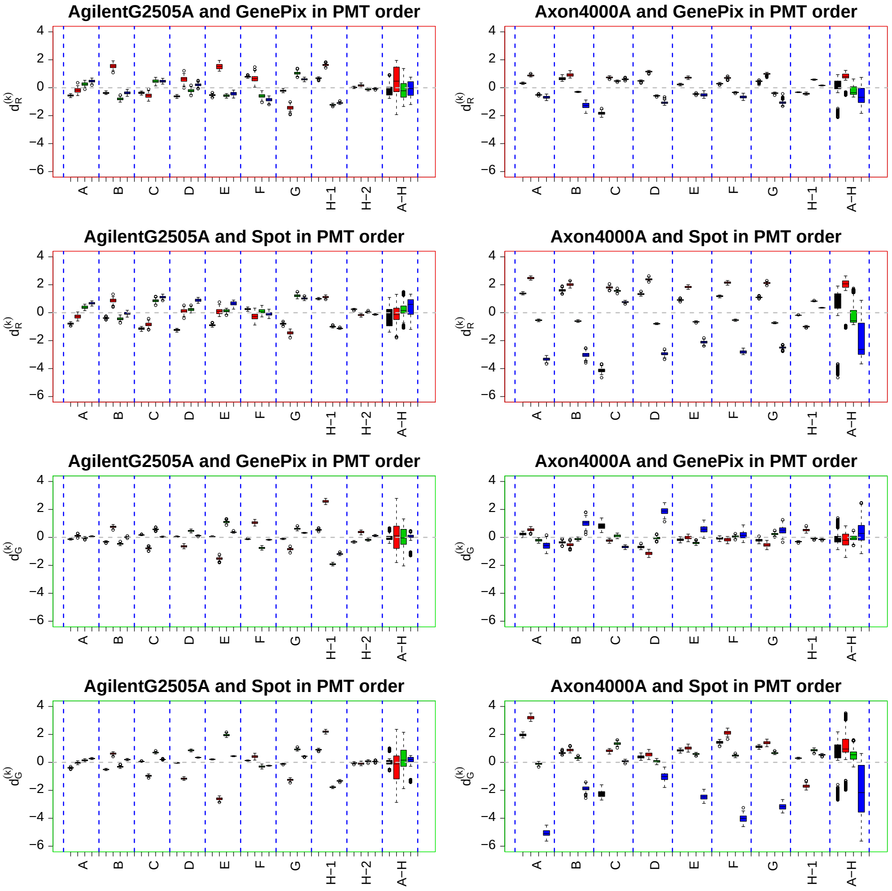


Figure 6: Estimated bias residuals $d_c^{(k)}$ for each array and for all arrays together from the Agilent scanner (left column) and the Axon scanner (right column). For each array the distribution of the four of $d_c^{(k)}$ are shown in increasing *PMT order*. For details on PMT settings, see Section 2.2.2. The top four graphs are for the red channel and the bottom four are for the green channel. For each channel the top two are estimates based on signals quantified by GenePix and the bottom two on signals quantified by Spot.

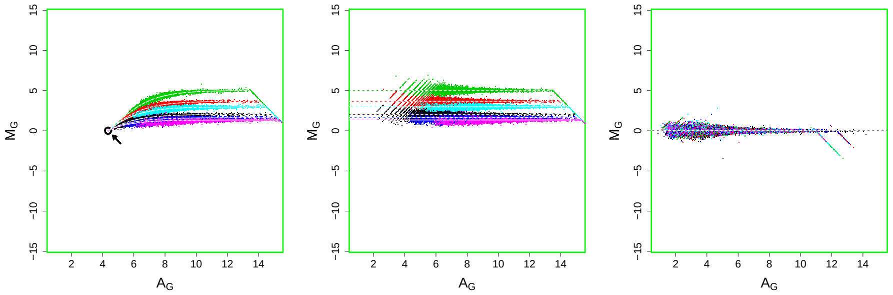


Figure 7: The affine transformation gives curvature in the M versus A plots, which is corrected for by the affine normalization method. The three scatter plots show the within-channel log-ratio versus the log-intensity for each of the six PMT pairs based on the same data as in Figure 1. *Left:* Observed signal for different PMT pairs. For *each* pair the estimated $(A_0, M_0) = (\frac{1}{2}\log_2(e_G^{(k)}e_G^{(l)}), \log_2(e_G^{(k)}/e_G^{(l)}))$ has been marked with a circle, cf. Figure 1. *Middle:* Bias corrected signals. *Right:* Bias and scale calibrated signals. The range of the M axis is twice the range of the A axis so that (12)-(13) appear as a rotation in the plots.

saturated signals, which can also seen in for instance Figure 7. These bumps are not visible if the saturated signals are removed.

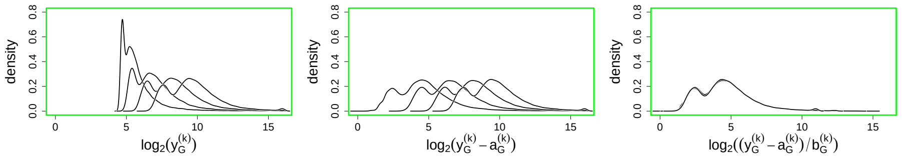


Figure 8: The affine normalization method makes the signal densities much more similar. *Left:* Density plots of the logarithm of the raw signals for each of the fours scans. *Middle:* Bias corrected signals. *Right:* Bias and scale calibrated signals. Data and colors are the same as in Figure 1.

3.2.1 Extended dynamical range

For the Agilent scanner the effective scale parameters $b_c^{(k)}/b_c^{(1)}$ were estimated to be in the order of approximately $1 : 3.5$. For the Axon scanner they were in the order of approximately $1 : 40$, cf. Table 1. Thus, the calibration method extends the effective dynamical range by a factor of 3.5 for data scanned on Agilent and a factor of 40 for data scanned on Axon. In both cases linearity was preserved.

4 Discussion

4.1 Sources of the bias

Because any bias introduced before the PMT detector would be registered and amplified differently by the PMT at different gains, we believe that the observed bias is due to the scanner and most likely its detector parts such as the analog-to-digital (A/D) converter after the PMT, but possibly also due to the image analysis method. The observed differences between the channels can be explained by the fact that there is one PMT and one A/D converter per channel, which may have slightly different properties. Although there are differences in bias between the two scanners, they are still of the same order, which we find remarkable. Another lab, which uses different arrays, reported biases also around 15-20 (personal communication). One possible reason for this is that the scanners could consist of similar parts although they are of different brands.

4.2 Other estimates

To rule out the obvious situation where *all* pixel intensities are biased, we compared the above estimates with the minimum pixel intensity. For example, for array A (scanned on Axon and analyzed with GenePix Pro), the *minimum pixel intensities* in the red channel were 9, 0, 8, and 9 for PMT 500, 600, 700 and 800 volts, respectively. In the green channel the minimum pixel intensity is 0 for all scans. It is not useful to use the *minimum spot signals*, $\hat{a}_c^{(k)} = \min_i y_{c,i}^{(k)}$, either. For example, for the above scan the average minimum signal across all scans in the red channel is 19.8 (median 19.5, std. dev. 0.96), but in the green channel it is 34.8 (median 28.0, std. dev. 19.6). The estimates for the green channels have a large variation compared to the estimates based on the affine model, cf. Table 3.

4.3 On background subtraction

If the scanner is the main source for the observed bias, then the background estimates should be affected by this bias as well and subtracting the background from the foreground estimates will therefore not only correct for physical background noise from the array itself, but also for the scanner bias. The *strong* intensity-dependent effects of the log-ratios that are due to the bias, are much less apparent if we apply background subtraction (not shown), giving more evidence to our hypothesis that the observed systematic effects originate from the scanner. Thus doing background correction might correct for the bias, but it will also introduce more noise at any given intensity (not depicted). Also, with the data set in hand *background subtraction* results in 4050 (7.3%), 6237 (11%), 7015 (13%) and 7349 (13%) negative values (in either channel), respectively whereas the *bias subtraction* results in no negative values. If we assume that the noise is additive such that the background is added to the foreground signals, then for probes with few or no fluorescent molecules the true foreground signal should be close or identical to the true background signal. As both are *estimates*, approximately half of the foreground signals for non-signal spots are less or equal to the corresponding background signals. Thus, about half of such spots results in negative signals. However, the different numbers of negative signals for different PMT voltages suggest that this cannot be the full explanation. One reason may be that the background *estimates* are likely to be biased [3]. An error model that incorporates different noise sources, but also different scan parameters, might give some answers to this. Some promising models in this context have already been suggested [17, 12, 8], but also other models based on empirical Bayes methods [15]. Another way to put it is that the background estimate is local and based on individual spots/pixels whereas the bias estimate is global, that is, there is one estimate for the whole array (although one could also imagine for instance spatially dependent estimates). Therefore, the background subtracted intensity estimates are noisier, resulting in more negative estimates for low intensity spots. Because remaining systematic effects due to the image analysis methods still have to be corrected for we find that the spot signals should be calibrated for biases.

4.4 Photo bleaching

Not considered in the above analysis is photo bleaching. We estimated the red dye (Cy5) to bleach about 2% and the green dye (Cy3) about 1% in a typical microarray experiment (not shown). Because the amount of bleaching is not too large, but also because it is a very complex phenomenon, we decided to not try to incorporate it in the above model. Some of the systematic variation seen in the bias estimates for the different PMT settings (Section 3.1.6) may be due to bleaching.

4.5 Signal density normalization

As shown in Section 3.2, the empirical distributions of signals match each other remarkably well after calibration. It is interesting to compare this method with the quantile normalization methods proposed by [6, 24]. The latter is based on the "statistical" assumption that the signals in all channels (scans) *should be equal* whereas as the former is based more on a "physical" assumption that the signals *should be linear* in the dynamical range. For a further discussion on this see [5].

4.6 Related work

Another method that combines multiple scans is the *masliner* (Microarray Spot LINEar Regression) algorithm [9]. It works by combining one low-PMT scan and one high-PMT scan into a new virtual scan. If a signal in the high-PMT scan is within a specified linear range its value is used, otherwise the corresponding signal from the low-intensity range is used after being transformed affinely to fit the high-PMT scan. To combine three or more scans, the new virtual scan can be combined with another PMT scan and so on. The result is that the effective dynamical range is extended. However, there are several unnecessary drawbacks. First, although several observations of the same spot concentration exist, which all may be within the dynamical range of the scanner, only one observation is used. Statistically, the average of all (calibrated) signals would be a more precise (less noisy) estimate. Second, since the scans are combined pairwise, the estimate of the affine relationship between the scans is less robust. Third, although a sensitivity discussion is carried out in the supplementary materials, masliner fits the affine models in a non-robust fashion (in L_2). Also, classical linear regression is used, which assumes no error in the explanatory variable. Since masliner makes the signals from different PMT settings *proportional to each other*, it will indeed remove for instance curvature in within-channel M versus A scatter plots. However, masliner does not model the possibility of a PMT-independent bias and will therefore not correct for it. We believe this is the reason why the authors observe a "curvilinear effect" [9, supplementary material]. For these reasons, we believe that the robust multiscan calibration method presented in this paper is superior to the masliner algorithm and should be used instead.

4.7 Incremental robust estimates

It turned out to be infeasible to estimate the model parameters based on *all pixel intensities*, which limited us to use only on a 10% subset of all data points. As argued above, pixel-based estimates are not reliable and therefore not of interest. However, for spot-based estimates the same limitations may apply as larger data sets are made available. We wish to overcome such memory constraints. For this reason, we investigate the possibility to use (approximative) incremental reweighted PCA methods [19, 16].

5 Conclusions

By scanning the same microarray at various PMT settings, we have shown that there exists a bias in the measurement of the concentration of fluorescent molecules in the spots on the microarray. Our analysis indicates that this bias is mainly due to the scanner, but also due to the image analysis methods. Moreover, by using a constrained affine model for the relationship between the obtained fluorescent intensities and fluorophore concentrations in the spots, we have been able to estimate the aforementioned bias. With estimates of the bias and scale parameters in each channel, back transformation gave estimate of the amount (up to a scale factor) of photons from each spot that enters the PMT, that is $x_{c,i} = h_c^{-1}(y_{c,i});\ \forall c, i$. Although not all photons originate solely from fluorophores in the target DNA, this is still a far better estimate of the amount of hybridized target DNA in each spot than the corresponding signal quantified by the scanner and the image analysis alone.

Before calibration data show a strong intensity-dependent effect in the log-ratios whereas after calibration there is no apparent intensity-dependent trend. Furthermore, the distributions of signals from subsequent PMT scans are almost identical after calibration. In addition to this, with multiple scans, the signal-to-noise ratio is increased. Finally, scanning at both low and high PMT settings extends the dynamical range of data, which gives higher resolution at low intensities without having to pay the price of saturated signals.

The proposed method can be applied to other microarray technologies such as single-channel oligonucleotide arrays or nylon arrays, and possibly to other gene-expression technologies such as quantitative real-time polymerase chain reaction (QRT-PCR). If an affine relationship is not observed, more sophisticated methods such as the promising quantile normalization methods suggested by [6, 21] may be used. More research is needed.

To conclude, we suggest that hybridized microarrays are scanned at two (preferably more) PMT gain levels to identify channel dependent bias terms. Knowing the exact PMT settings is not important, but it can be shown that the larger the differences are the more precise the estimates will be. We recommend that the scans are done in decreasing PMT-gain order (although we did not do so here). Given the estimates, data can then be calibrated easily.

For two-channel microarrays, after calibrating each channel separately, a similar strategy can be applied once more to bring differently labeled channels to the same scale as suggest in [5]. This would rely on the assumption that the amounts of *hybridized DNA* in all channels are approximately equal for the majority of the spots, which in turn is based on the commonly used assumption that most genes are non-differentially expressed. This also applies to normalization between arrays.

All necessary methods are made available in a free R package named *aroma* [4]. A typical usage is `calibrateMultiscan(rg)` where `rg` is the object containing the red and green signals. In addition, we are currently implementing the methods as a plug-in module for the BASE system [18].

Acknowledgments

We wish to thank Professor Ola Hössjer at Mathematical Statistics at Stockholm University for his most valuable feedback on the methods and the manuscript. We also thank Professor Åke Borg at Department of Oncology, Lund University for providing microarray data. This work was supported by grants from the Swedish Foundation for International Cooperation in Research and Higher Education (STINT), Fulbright Commission, Foundation Blanceflor Boncompagni-Ludovisi née Bildt, Royal Swedish Academy of Sciences, Royal Physiographic Society in Lund, Swedish Cancer Society, Ingabritt and Arne Lundberg's Research Foundation, and American Cancer Society. Microarrays were produced by the SWEGENE DNA Microarray Resource Center at the BioMedical Center B10

in Lund, supported by the Knut and Alice Wallenberg foundation through the SWEGENE consortium.

References

[1] Agilent Technologies, Inc., Palo Alto, CA. *Agilent G2565AA and Agilent G2565BA Microarray Scanner System - User Manual*, third edition, September 2002. G2566-90007.

[2] CSIRO Australia. *Spot user's manual.* CSIRO Mathematical and Information Sciences, Image Analysis Group, July 2003. http://spot.cmis.csiro.au/spot/.

[3] Anders Bengtsson. Microarray image analysis: Background estimation using region and filtering techniques. Master's Theses in Mathematical Sciences, Mathematical Statistics, Centre for Mathematical Sciences, Lund Institute of Technology, Sweden, December 2003. 2003:E40.

[4] Henrik Bengtsson. aroma - An R Object-oriented Microarray Analysis environment. Preprint in Mathematical Sciences 2004:18, Mathematical Statistics, Centre for Mathematical Sciences, Lund University, Sweden, 2004.

[5] Henrik Bengtsson and Ola Hössjer. Methodological study of affine transformations of gene expression data with proposed normalization method. Preprints in Mathematical Sciences 2003:38, Mathematical Statistics, Centre for Mathematical Sciences, Lund University, Sweden, 2003. Submitted.

[6] B.M. Bolstad, R.A. Irizarry, M. Åstrand, and T.P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–93, Jan 2003.

[7] James P. Brody, Brian A. Williams, Barbara J. Wold, and Stephen R. Quake. Significance and statistical errors in the analysis of DNA microarray data. *Bioinformatics*, 99(20):12975–12978, October 2002.

[8] Xiangqin Cui, M. Kathleen Kerr, and Gary A. Churchill. Data transformations for cDNA microarray data. Technical report, The Jackson Laboratory, USA, 2002.

[9] Aimée M. Dudley, John Aach, Martin A. Steffen, and George M. Church. Measuring absolute expression with microarrays with a calibrated reference sample and an extended signal intensity range. *Proc Natl Acad Sci U S A*, 99(11):7554–9, May 2002.

[10] Sandrine Dudoit, Yee Hwa Yang, Matthew J. Callow, and Terence P. Speed. Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments. Technical Report 578, Department of Statistics, University of California at Berkeley, 2000.

[11] David J. Duggan, Michael Bittner, Yidong Chen, and Paul Meltzer & Jeffrey M. Trent. Expression profiling using cDNA microarrays. *Nature Genetics*, 21(1 Supplement):10–14, January 1999.

[12] Wolfgang Huber, Anja von Heydebreck, Holger Sültmann, Annemarie Poustka, and Martin Vingron. Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 1:1–9, 2002.

[13] Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

[14] Burble Industries Inc. *Photomultiplier Handbook*, 1980.

[15] Charles Kooperberg, Thomas G. Fazzio, Jeffrey J. Delrow, and Thoshio Tsukiyama. Improved background correction for spotted DNA microarrays. *Journal of Computational Biology*, 9:55–66, 2002.

[16] Y. Li, L-Q. Xu, J. Morphett, and R. Jacobs. An integrated algorithm of incremental and robust pca. In *IEEE International Conference on Image Processing (ICIP2003)*, September 2003.

[17] David M. Rocke and Blythe Durbin. A model for measurement error for gene expression arrays. *Journal of Computational Biology*, 8(6):557–569, 2001.

[18] Lao H. Saal, Carl Troein, Johan Vallon-Christersson, Sofia Gruvberger, Åke Borg, and Carsten Peterson. BioArray Software Environment (BASE): a

platform for comprehensive management and analysis of microarray data. *Genome Biol*, 3(8):SOFTWARE0003, Jul 2002.

[19] D. Skocaj and A. Leonardis. Weighted and robust incremental method for subspace learning. In *ICCV03*, pages 1494–1501, 2003.

[20] R Development Core Team. R Language Definition (v1.7.1, draft), June 2003.

[21] Natalie Thorne. *Microarray analysis*. PhD thesis, University of Melbourne, January 2004.

[22] Yee Hwa Yang, Michael Buckley, Sandrine Dudoit, and Terry Speed. Comparison of methods for image analysis on cDNA microarray data. Technical Report 584, Department of Statistics, University of California at Berkeley, Nov 2000.

[23] Yee Hwa Yang, Sandrine Dudoit, Percy Luu, and Terence P. Speed. Normalization for cDNA microarray data. In Michael L. Bittner, Yidong Chen, Andreas N. Dorsel, and Edward R. Dougherty, editors, *Proceedings of SPiE*, volume 4266 of *Microarrays: Optical Technologies and Informatics*, pages 141–152, San Jose, California, June 2001. The International Society for Optical Engineering.

[24] Yee Hwa Yang and Natalie P. Thorne. Normalization for two-color cDNA microarray data. In Darlene R. Goldstein, editor, *Science and Statistics: A Festschrift for Terry Speed*, volume 40 of *Monograph Series*, pages 403–418. IMS Lecture Notes, 2003.

F

Paper F

aroma - An R Object-oriented Microarray Analysis environment

Henrik Bengtsson

Mathematical Statistics at the Centre for Mathematical Sciences, Lund University, Box 118,
SE-221 00 Lund, Sweden. Email: hb@maths.lth.se.

ABSTRACT

We have implemented a self-contained package for DNA microarray analysis in R. The package is named aroma and is formerly known as com.braju.sma. The purpose of the package is to provide end-users with easy access to the latest statistical tools and methods, to provide other statisticians with a structured platform to develop and test new analysis methods, and to make our methods available to an audience as broad as possible. In turn, this give us most valuable feedback for improving the package and its algorithms. The package provides methods for low-level analysis and pre-processing of spotted microarray data with two or more channels, although the recent methods are applicable to single-channel microarrays as well. Many of the commonly known and widely used normalization methods have been implemented as well as our own normalization and calibration methods in both early and mature versions. A few high-level analysis methods do exist. Import and export of microarray data is supported for most free and commercial file formats. Support for new ones can easily be added. Connectivity to other microarray analysis platforms based on relational databases is planned. For exploratory data analysis, the package provides a large set of methods for visualization of data with automatic graphical annotations. Textual and symbolic annotation of graphs is straightforward. Generated graphics can be exported for on-screen presentations and high-resolution printed publications. Report generators for simultaneously creating documents in plain text, HTML, and LaTeX are available, making it easy to create batch analysis scripts. Detailed documentation with code examples and references to the literature are included. In order to provide time and memory efficient algorithms, an object-oriented design and implementation utilizing reference variables has been used. In addition, to minimize the risk for programming mistakes, a consistent application protocol interface (API) was developed by enforcing an R coding convention. This will make it possible for a new user to quickly become familiar with the structure of classes and methods enhancing overall productivity. To further minimize the risk for programming errors, method arguments are validated as far as possible and explanatory error messages are used. Priority is also on backward compatibility with previous versions, compatibility with other packages, and potential future packages. The package does not provide a graphical user interface (GUI) per se, but the object-oriented API and the fact that reference variables are used make it easy for third-party developers to implement a GUI for, say, everyday high-throughput normalization. The package is available online as open source written in 100% R for maximum accessibility on all platforms. Installation is straightforward.

Keywords: microarray analysis; single-, two-, and multiple-channel microarrays; low-level analysis; pre-processing of data; visualization; calibration; normalization; data import and export; object-oriented programming; reference variables; coding conventions; data hiding; encapsulation; virtual fields.

1 Introduction

Aroma is a package for *low-level analysis* (calibration and normalization) of two- or multiple-channel microarray data, but part of it may as well be applied to single-channel microarray data. The package is implemented in pure R [13]. This means that it runs out of the box and works identically on operating systems such as Windows, Mac OS X, Sun Solaris, and any Linux system.

It is not our intention of this paper to give a complete description of the aroma package, but to give an general overview and to provide enough details to become familiar with the structure of the package and its methods. Hopefully, the reader will be tempted enough to install the package and start use it in his or her own microarray analysis.

The outline of the paper is as follows. In Section 2, we give a brief introduction to object-oriented programming and reference variables in general, and emphasize the fact that the aroma packages pass variables by reference and not by value (or copy), as is the default in R. Section 3 explains how microarray signals are stored and queried. After these introductory sections, a walk-through of the aroma package is provided. After describing how R and the package is started, it is described how typical microarray data generated by common academical and commercial microarray image analysis applications is imported and how foreground and background signals are extracted from such. A few of the methods for generating graphical representation of microarray data are then presented. In the following subsection, calibration and normalization is covered. The walk-through is ended by a short description on how normalized data is written to file. In the section 5, some additional useful features are described. Report generators are introduced, the help and documentation of the package is described, and compatibility with other packages and other scientific applications is discussed. The philosophy of the design and implementation of the package is given in Section 6. This part is recommended to anyone who wish to extend aroma with new classes or methods, but may be skipped by anyone not familiar with software development. Installation instructions are given in 7.

2 Brief about object orientation and reference variables

For the novel user we give a very brief introduction to the object-oriented design used by us and reference variables as these two concepts are fundamental to the aroma package and might differ slightly from what is otherwise common in R. For a general introduction to R, see [15].

2.1 Object orientation

In object-oriented programming two concepts are of major importance, namely *classes* and *objects*. Simply speaking, a *class* describes and defines a structured data type, for instance, a data table containing signals and other spot attributes returned by the Axon GenePix software. An *object* is then an *instance* of a class, meaning that it contains the actual data on which analysis is to be done. Several instances can exists of the same class. A simple rule for understanding this is that a class exists as source code at all time whereas an object exists while running the analysis. Moreover, different classes (data structures) for different purposes may be defined. A more real-world analogy is on cooking. In a cookbook there are recipes. These recipes describe and define certain types of dishes and can be thought of as classes. For any given recipe one or several meals (of identical type) can be prepared. These can be thought of as the objects, that is, they are instances of the recipe.

Different data structures are often more or less related to each other. For instance, a class for Axon GenePix microarray data is related to a class for QuantArray microarray data by the fact that they both represent quantified signals for microarray data. In contrast, these two classes are not related to a class describing, say, the heart beat of a mouse. Lets denote the two classes by *GenePixData* and *QuantArrayData*, respectively. A common *superclass* of these two classes would then be *MicroarrayData*, describing the fact that the two classes share similar properties (from the superclass). For instance, they both contain foreground and background estimates, but in addition have properties that are unique to each of them. We say that the *GenePixData* class *extends* (the properties) or *inherits from* the *MicroarrayData* class. Furthermore, we say that *GenePixData* is a *subclass* of the *MicroarrayData* class.

To continue our analogy to cooking, the recipes in a cookbook are often categorized into sections such as starters, main dishes, and desserts. These categories may be though of as *abstract* (super-)classes of the actual dishes. Simply, you cannot ask someone to make just a dessert (the superclass), but you have to be more specific and say which type of dessert (the class) in order for someone to get the exact ingredients, make the dessert, and bring it to you (the object). In our microarray example, the class *MicroarrayData* is abstract, because it is "not specific enough". The reason for having a *hierarchy of classes* in this way is to make it simpler for both the programmer and the user to become familiar with, and navigate between, all the data structures and their properties. For further details on object-oriented programming in R, see [11, 12, 4].

2.2 Reference variables

The language in R is a so called *functional language*, which means that all methods (functions) take some input objects, performs calculations and operations based on these and returns a *new* object. This imitates the concept of a function in mathematics, which is the basic language for all scientific analysis. The main advantage is that functions cannot modify an object (just return a new one) leaving no side effects or surprises to the user. This has shown to be very useful. The disadvantage is that it is less memory efficient since often two copies of the same data structure will live in the memory at the same time. This is of huge concern in microarray analysis because of the large amount of data processed. For instance, consider an *RGData* object containing red and green foreground signals for thousands of spots. Internally, R will then, for a function that subtracts background signals from these signals, first create a copy of the object containing all foreground signals and *then* subtract the background signals on this new object, which is then returned. In the aroma package we have chosen another strategy; we use *reference variables* [4]. The difference is basically that a function, or a *method* to differentiate it from the original meaning of a function, can change the values of any object passed to it. We say that objects are *passed by reference*, contrary to being *passed by value or copy*.

There are other advantages of using pass by reference than just being memory efficient. Because a method can *change the state* of an object, it can store information that succeeding analysis methods use. For instance, the user may specify that all signals for a certain set of genes in a specific data set (object) should be highlighted in red. This "request" will be stored in the object and recognized by all future plot methods that will highlight the data points without the user having to specify which data points to highlight in every call[14]. See `?Filter` for more details. The `highlight()` method described below is another example that makes use of reference variables.

We ask the reader to keep in mind that aroma uses reference variables.

3 Representation of data

Microarray data in aroma is represented by objects of classes that are all subclasses of the *abstract* class *MicroarrayData*. Any such class contains probe (spot) signals from one or several microarrays. The signals are stored in a field that is a $I \times K$ matrix, where I is the number of probes and K the number of arrays. Thus, the signals in row i are the signals for probe i on all $k = 1, \ldots, K$ arrays, and the signals in column k are the signals from array k and all probes $i = 1, \ldots, I$. If a *MicroarrayData* object[15] contains signals from more than one channel, each

[14] This can be done with functional programming too, but is much less memory efficient.

[15] When we refer to a "*MicroarrayData* object" we actually mean an object of a class that extends the *MicroarrayData* class.

channel is stored in a separate field. For instance, the *RGData* class is a basic class that represents two-channel microarray data with two matrix fields R (red) and G (green) of equal dimensions. Consider an object rg of class *RGData*. Then the signal in the red channel for spot $4, 5, 14 - 20$ on the first arrays is obtained by

```
rg$R[c(4,5,14:20),1]
```

and the same spots on all other arrays by

```
rg$R[c(4,5,14:20),-1]
```

Note that field R of object rg is accessed by the $-operator similar to how elements in lists are accessed. Another example is to calculate the log-ratios and log-intensities [20]

$$\begin{aligned} M &= \log_2 \frac{R}{G} \\ A &= \frac{1}{2} \log_2(RG) \end{aligned} \tag{1}$$

for all probes and arrays. M is for "minus" and A is for "add", because the ratios and products become subtraction and addition under the logarithm. In R this can be done by

```
M <- log(rg$R/rg$G, base=2)
M <- log(rg$R*rg$G, base=2)/2
```

which will create a new $I \times K$ matrix. Indeed, since these transforms are so common, it is not necessary to do the above every time. For any *MicroarrayData* class for two-channel data, the log-ratios and log-intensities can be obtained by rg$M and rg$A, respectively. For instance,

```
rg$M[c(4,5,14:20),-1]
rg$A[c(4,5,14:20),-1]
```

These fields are in fact not stored in the object, but are *calculated on-demand* and do therefore not take up valuable memory. This is unique to aroma. For more details, see Section 6.4 on virtual fields. Continuing, to assign new values to some probes do

```
rg$R[14:20,1:2] <- 0
```

which will zero the signals in the red channel for probe $14 - 20$ on array one and two. The *MAData* class is complementary to the *RGData* class, in the meaning that it stores the log-ratios and the log-intensities, and calculates the red and the green signals on-demand by the inverse transformations

$$\begin{aligned} R &= \sqrt{2^{2A+M}} \\ G &= \sqrt{2^{2A-M}}. \end{aligned} \tag{2}$$

Conversion between these two classes is done by `as.MAData()` and `as.RGData()`. For instance,

```
ma <- as.MAData(rg)
rg2 <- as.RGData(ma)
```

will create an *MAData* object `ma` from to the *RGData* object `rg` and then another *RGData* object `rg2`. From the above discussion (assuming no non-positive signals), all of `rg$M`, `ma$M`, and `rg2$M` will return equal values. Moreover, objects `rg` and `rg2` are equal, which can be tested by[16]

```
> equals(rg, rg2)
[1] TRUE
```

Note that these are two different objects, making `identical(rg, rg2)` to return `FALSE`. Compare this with

```
> rg3 <- rg
> identical(rg, rg3)
[1] TRUE
```

This is because `rg` and `rg3` *refer* to the same object (instance). At this point we have to admit that we have been sloppy with the words. When we say "the object `rg`" we actually mean "the object that the reference variable `rg` refers to", but having to do this each time would be too lengthy and we will allow ourselves to use the shorter form whenever it is non-ambiguous to do so. Thus, the references variables `rg`, `rg2`, and `rg3` refer to two objects only. Hence, the following outcome:

[16] We will display the R prompt '>' whenever both commands and results are shown in the same code example, otherwise not.

```
> rg3 <- rg
> rg$R[1,1] <- 2
> rg3$R[1,1]
[1] 2
```

To further emphasize the difference between objects and references, to query the size of an *object* use

```
> objectSize(rg)
[1] 48716
```

and compare this with the built-in R function

```
> object.size(rg)
[1] 452
```

which will return the size of a *reference variable* itself, which is only a few hundred bytes large regardless of the size of the object it is referring to.

4 A walk-through of the package

The R software is started by typing

```
R
```

at the command prompt. On Windows platforms, the *Rgui* environment may also be used. When in R, the aroma package is loaded by

```
library(aroma)
```

giving output similar to

```
Loading required package: R.basic
Loading required package: R.oo
R.oo v0.52 (2004-07-01) was successfully loaded.
Loading required package: R.io
R.io v0.52 (2004-07-01) was successfully loaded.
Loading required package: R.graphics
Loading required package: R.colors
R.colors v0.52 (2004-07-01) was successfully loaded.
R.graphics v0.52 (2004-07-01) was successfully loaded.
```

which confirms that it is loaded together with all necessary packages. For a first test-run, try

```
example(GenePixData)
```

To quit R, use `quit()`. For installation instructions, see Section 7.2.

4.1 Importing microarray data

This section briefly explains how to read data into aroma. The package provides methods to read and write microarray data that is stored in one of the many file formats generated by some of the most well-known cDNA microarray image analysis applications such as GenePix, ImaGene, QuantArray, ScanAlyze, Spot [1, 19], and Spotfinder. Most versions of these applications are supported. Consider an example where three Axon's GenePix Result (GPR) files named "slide1.gpr", "slide2.gpr", and "slide3.gpr" are stored in the current working directory. Any one of these files may be read by the *static* method `read()` of class *MicroarrayData*;

```
gpr <- MicroarrayData$read("Slide1.gpr")
```

Static (class) methods are methods that are specific to a class, but not necessarily to an instance of it. This is typical for read methods for which an object does not exist prior to the call. To call static methods, the `$` operator is used, indicating that the method belongs to a class much like a field belongs to an object, or an element belongs to a list. The above method will return an object of a subclass to *MicroarrayData* (recall that this class is abstract) that is specific for the data set read. Among the commonly used ones, aroma defines classes such as *GenePixData*, *ImaGeneData*, *QuantArrayData* etc. to represent the different data formats.

In addition to this, several files may be read at once by either giving a *vector of filenames*, or by specifying a *string pattern* matching filenames, to be read;

```
gpr <- MicroarrayData$read(pattern="*.gpr")
```

which will read (in alphabetic order) all files that ends with "gpr" into one *GenePixData* object. For more details on filename patterns, see R help on *regular expressions*, e.g. `?regexp`.

In the above two examples, the read method returns a reference, which we denote by `gpr`, to an object of class *GenePixData* (a subclass of *MicroarrayData*, as can be seen if we write `print(gpr)` or, shortly, by typing `gpr` at the command prompt. For the latter example we get;

```
> gpr
[1] "GenePixData: Number of slides: 3. Number of fields: 43. Layo
ut: Grids: 4x4 (=16), spots in grids: 18x18 (=324), total number
of spots: 5184. Spot names are specified. Spot ids are specified."
```

From this we see that the object is of class *GenePixData* and contains 43 different measurements from 3 microarrays (slides) with 5184 spots. The array was printed with a 4x4 print-tips head etc. To investigate the contents of *the object* further

```
names(gpr)
```

will return the names of all 43 fields, and

```
ll(gpr)
```

will return a detailed list of all field names, their types and their sizes etc. Here is an excerpt:

```
           member data.class dimension object.size
1    % > B532+1SD     matrix c(5184,3)       63000
2    % > B532+2SD     matrix c(5184,3)       63000
3    % > B635+1SD     matrix c(5184,3)       63000
4    % > B635+2SD     matrix c(5184,3)       63000
5        B Pixels     matrix c(5184,3)       63000
6       B532 Mean     matrix c(5184,3)       63000
7     B532 Median     matrix c(5184,3)       63000
...
```

Alternatively to the $-operator, fields can be accessed through the [[-operator as follows:

```
gpr[["% > B635+2SD"]][c(4,5,14:20),1]
```

which again is similar to how list structures in R work. To know what methods are available for a given *class*, print(GenePixData), or shorter just the class name typed at the R prompt, will give a summary of the class;

```
GenePixData extends MicroarrayData, Object {
  public layout
  public version
  public anonymize(...)
  public append(other)
  public getAbscent(slides=NULL, include=NULL, ...)
  ...
}
```

Like for any class or method, further information can be found in the help pages installed with the package, e.g. help(GenePixData) or shorter ?GenePixData. For more details about the help facilities in aroma, see Section 5.2.

4.2 Extracting the foreground and background signals

As seen in previous sections, data output by microarray image analysis software contains a large number of measurements per probe, but in most analysis it is only the *foreground and background signals* that are used. Because of this, the *RawData* class exists. The estimates of the foreground and the background signals in a *GenePixData* object can be extracted by

```
raw <- getRawData(gpr)
print(raw)
[1] "RawData: R (5184x3), G (5184x3), Rb (5184x3), Gb (5184x3), L
ayout: Grids: 4x4 (=16), spots in grids: 18x18 (=324), total numb
er of spots: 5184. Spot names are specified. Spot ids are specifi
ed."
```

This object contains 5184 foreground signals R and G, and 5184 background signals Rb and and Gb from all three arrays. Note how the array layout is included automatically. A *GenePixData* object contains a few different types of foreground and background estimates. By default, the median pixel values are returned. To retrieve other estimates, see ?GenePixData. Continuing, if the other measurements in the *GenePixData* object is not needed anymore, it can be deleted by standard R procedures, that is

```
rm(gpr)
```

If no other reference variables refer to the same object, the object will be deallocated by the built-in R *garbage collector*. Thus, even if reference variables are used, a user or a programmer does not have to worry about memory management. Note the decrease in memory usage;

```
> objectSize(raw)
[1] 258416
> objectSize(gpr)
[1] 5374740
> objectSize(raw)/objectSize(gpr)
[1] 0.04807972
```

Next step in a typical microarray analysis is to perform background correction or not. This is done by one of the following two alternatives:

```
ma <- getSignal(raw, bgSubtract=TRUE)
ma <- getSignal(raw, bgSubtract=FALSE)
```

4.3 Visualization

In microarray analysis, as well as all other statistical analysis, it is most useful to initially display data in as many ways as possible. This will show the nature of data and reveal problems such as systematic effects in measurements etc. In addition to the huge number of exploratory data analysis (EDA) methods available in R in general, the aroma package provides a large set of graphical EDA methods that are special for microarray data. They have been designed such that they require minimal "interaction" by the end user. For instance, axes are labelled and scaled automatically, color annotation is automatic, and highlighting of certain data points is made simple. Here are two examples:

```
plotSpatial(gpr)
plotSpatial(ma)
```

Both of these commands will create a spatial plot of the 5184 observed log-ratios that by default are colored in a red-to-green and dark-to-bright scale. Other color scales may be chosen and color filters are available. Since a *GenePixData* object contains information about the spot positions and spot diameters, the first command will display the signals as discs of different sizes as shown in the left panel of Figure 1. For the second command no spot positions or sizes are available, but only the array layout, resulting in the image in the right panel.

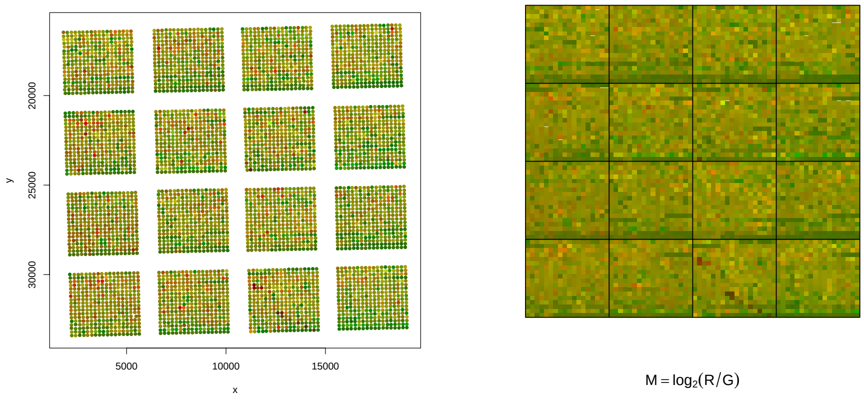


Figure 1: Spatial plots of two-color microarray data. *Left:* Spatial plot from a *GenePixData* object. *Right:* Spatial plot for an *MAData* object with only array layout information.

For all *MicroarrayData* classes a generic `plot()` method is defined. Depending on which class the object belongs to, different types of plots will be created. For instance, calling `plot(rg)` on an *RGData* object `rg` will generate a scatter plot where the red signals are plotted against the green signals. Similarly, for an *MAData* object a log-ratio versus log-intensity plot will be generated. Moreover, the generic `highlight()` method will highlight certain spots in any type of plot generated by an aroma plot methods. For example, in

```
idx <- (abs(ma$M) > 1.5)
plot(ma)
highlight(ma, idx, col="blue")
rg <- as.RGData(ma)
plot(rg)
highlight(rg, idx, col="blue")
```

spots with an absolute log-ratios above 1.5 are highlighted. The outcome is as in Figure 2. Note, if the `highlight()` method would be called after, say, `plotSpatial()`, the spots would be highlighted in the spatial plots. For the interested, this is another example where reference variables have been used; the objects remember what the last plot call was and this information is used when the highlighting is done.

In addition to spatial and X-Y scatter plots, there are various other plot method such as print-order [2], dip-order [2], and box-and-whisker plots. For example,

```
plotDiporder(ma)
boxplot(ma, groupBy="printtip")
```

with results as in Figure 3. Additional plot styles will be given in the following sections. For a complete list of plot methods, see the help pages.

4.4 Calibration and normalization

Calibration and normalization of DNA microarray data is about removing artifacts in the observed signals that are due to systematic variations in the microarray process and not due to biological variation, which is the variability of interest. Removing systematic effects is necessary in order for higher-level analysis such as identification of differentially expressed genes and gene classification to be correct.

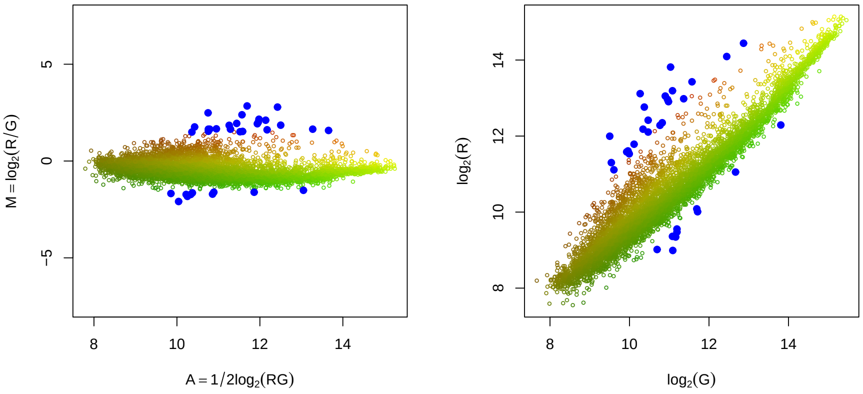


Figure 2: Examples of `plot()` and `highlight()`. *Left:* Default scatter plot for an *MAData* object. *Right:* Default scatter plot for an *RGData* object. In both graphs, the same genes have been highlighted in blue.

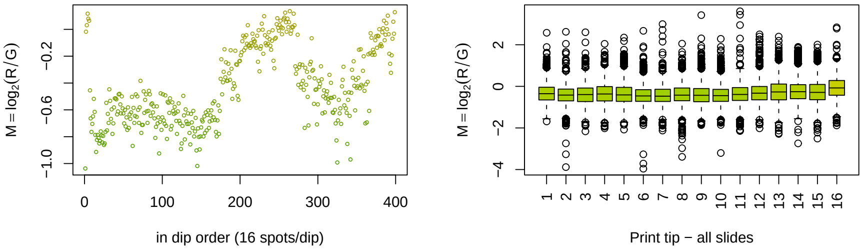


Figure 3: *Left:* Examples of a dip-order scatter plot of the average of the 16 log-ratios from all 384 print-dips in the order the corresponding spots were printed on the arrays. *Right:* A box-and-whisker plot of the log-ratios on all arrays stratified by print-tip.

For a formal definition of the difference between calibration and normalization, see the introduction of [5]. Here is an example of a curve-fit normalization

```
plot(ma)
normalizeCurveFit(ma)
plot(ma)
```

Note, normalization methods in aroma will update the object passed to it. This will save memory, but it means that the non-normalized data will be lost. If this is not wanted, the object can be *cloned* before calling the normalization method;

```
maNormalized <- clone(ma)
normalizeCurveFit(maNormalized)
```

To normalize data such that the scales of the log-ratios becomes equal for all arrays, so called across-slide (scale) normalization can used;

```
normalizeScaleAcrossSlides(ma)
```

Note that this should not be necessary for the quantile and the affine normalization methods that are described next. Affine normalization [7], which is performed in the (G,R)-signal space contrary to the (A,M) space, is done by

```
normalizeAffine(rg)
```

The effect of the affine normalization in the M versus A plane is depicted in Figure 4. As a final example, quantile normalization [10] is done by

```
normalizeQuantile(rg)
```

Quantile normalization will normalize data in each separate channel such that the empirical densities of the signals from all arrays become as similar as possible. See Figure 5, which compares the density functions from six arrays normalized in three different ways.

Several of the normalization methods can be applied spatially, in one dimension such as the smooth print-order normalization, or by stratifying on gene or spot group, e.g. by print-tip, microtiter plate, clone library etc. For instance, to apply affine normalization on each print-tip individually, do

```
  normalizeAffine(rg, groupBy="printtip")
```

and similar for the other normalization methods.

When several scans of the same array are available, the robust affine *multiscan calibration* method described in [8] can be utilized. It will correct for scanner biases, which will remove intensity-dependent effects due to the scanner, and calculate the average signals so that the scanner's effective dynamical range is extended and

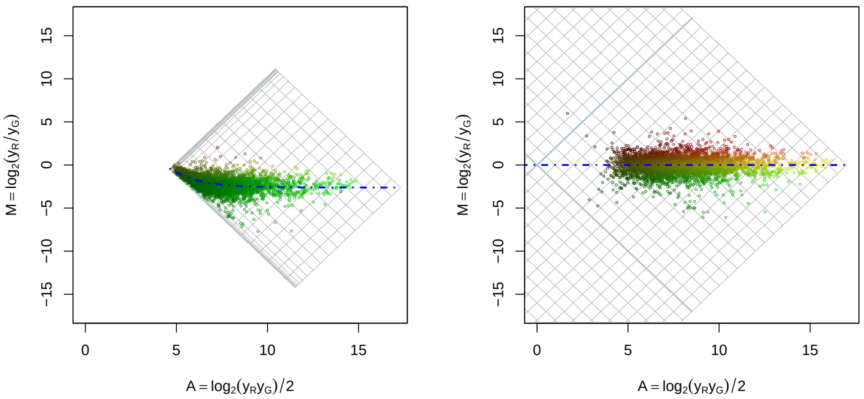


Figure 4: *Left:* Observed log-ratios versus log-intensities. The overlaid grid is the rotated $(\log_2 G, \log_2 R)$-grid as it is after being deformed by (estimated) affine transformation. The left corner corresponds to $(G, R) = (1, 1)$ and the dashed curve is the where the normalized data satisfies $G = R$. *Right:* The affinely normalized signals with grid deformations removed.

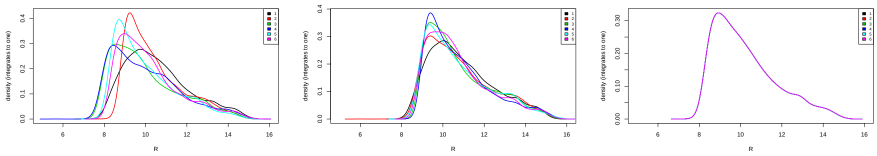


Figure 5: Density plots for the signals (logarithm base 2) in the red channels from six different arrays after various normalization methods (the labels on the axes read "R" (red channel) and "density (integrates to one)"). *Left:* Curve-fit normalization and across-slide scale normalization. *Middle:* Affine normalization. *Right:* Quantile normalization.

the signal-to-noise level is increased. This is illustrated in Figure 6. Multiscan calibration, which should be applied before other normalization methods, is done by

```
calibrateMultiscan(rg)
```

where the columns of the *RGData* fields represent signals from multiple scans of the same array. Multiscan calibration is applied to each channel separately. For further details and for all available calibration and normalization methods, see the aroma help pages.

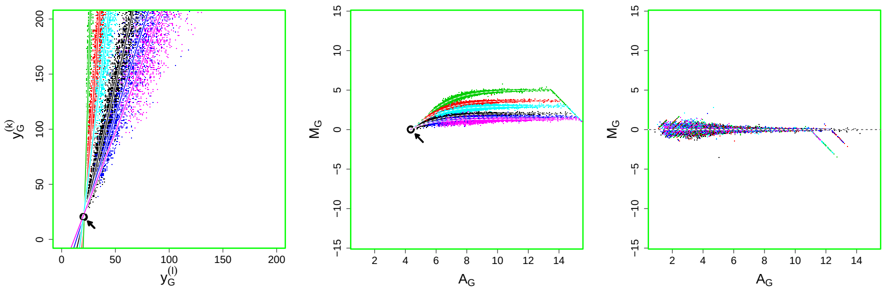


Figure 6: Example of a multiscan calibration of four scans from the same array. *Left:* Green versus green signals for all six possible pairs from the four different scans done at various sensitivity settings. *Middle:* The same data plotted as within-channel M vs A plots. *Right:* The same but now after calibration. Intensity-dependent effects have been removed and the signals can be averaged to decrease noise levels.

4.5 Exporting microarray data

Normalized microarray data can be exported for further analysis in other applications or to be saved to file for future usage. To save a *MicroarrayData* object to a tab-delimited file do

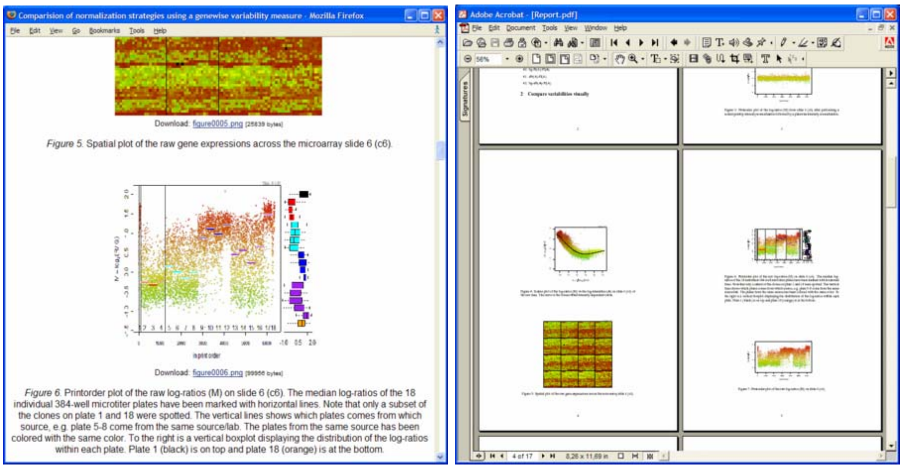
```
write(ma, "ma.normalized.dat")
write(rg, "rg.normalized.dat")
```

and similar for objects of other classes.

5 Miscellaneous features

5.1 Report generators

For everyday analysis, a user will most likely write so called *batch scripts* that will perform many steps automatically. To support such batch jobs, a set of *Reporter* classes that generate documents containing text, tables and figures, have been developed. Reports in plain text (no graphics), in HTML, and in LaTeX, are generated by the *TextReporter*, *HtmlReporter* and *LaTeXReporter* classes, respectively. For the records, these classes are implemented by the *R.io* package (part of the R.classes bundle), which is required (see Section 7.2) and loaded when aroma is loaded. Because all *Reporter* classes implement the same API, generating figures (of appropriate format), tables, lists, code examples, table of contents etc. is done in a uniform way regardless of the output format. Reports of multiple formats can be generated in parallel by utilizing the *MultiReporter* class. For an example of a microarray analysis report generated by the above classes, see Figure 7. As a complement to *Reporter* classes, Sweave [14], which is part of the tools package, can be used to write a hybrid of LaTeX and R documents. Such documents may contain R code that generate figures, tables and other visual output, which is automatically inserted into the generated LaTeX document. The main difference between Sweave and the *Reporter* classes is that with the former, the report is written as a document from which the R code is extracted and evaluated whereas with the latter it is the R script that generates the document. Both can be used together.


Comparision of normalization strategies using a genewise variability measure - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
M = log2(R/G)
printorder
Adobe Acrobat - [Report.pdf]
File Edit Document Tools View Window Help
Signatures
2 Compare variabilities visually
4 of 17
8.26 x 11.69 in

Download: figure0005.png [25839 bytes]

Figure 5. Spatial plot of the raw gene expressions across the microarray slide 6 (c6).

Download: figure0006.png [99966 bytes]

Figure 6. Printorder plot of the raw log-ratios (M) on slide 6 (c6). The median log-ratios of the 18 individual 384-well microtiter plates have been marked with horizontal lines. Note that only a subset of the clones on plate 1 and 18 were spotted. The vertical lines shows which plates comes from which source, e.g. plate 5-8 come from the same source/lab. The plates from the same source has been colored with the same color. To the right is a vertical boxplot displaying the distribution of the log-ratios within each plate. Plate 1 (black) is on top and plate 18 (orange) is at the bottom.

Figure 7: Example of an HTML page (*left*) and a LaTeX document (*right*) comparing different normalization methods. These were generated in parallel by an aroma script utilizing the *MultiReporter* class.

5.2 Help, manuals and other documentation

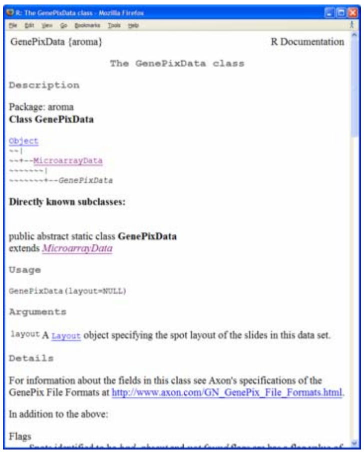
The aroma package is continuously updated with new algorithms and visualization methods. For this reason, the package's help documentation, which comes with all installations and is updated equally often, should always be queried in the first place. The HTML version of these help pages can be accessed by typing `help.start()`, clicking the "Packages" link followed by the "aroma" link. At the top of this root help page for aroma there is a list of links to separate sections, e.g. "Introduction", "Classes", "Import and Export of Data", "Normalization" and so on. All public methods and functions of the package are listed in these sections. Note that each method is also accessible via the page for the class for which it is coupled to. The "Classes" section lists all classes in the package with links to detailed help for each of them. For several classes and methods complete examples are given, examples which can be tried either by cut'n'paste or by

```
example(GenePixData)
```

It is good to know that all examples are asserted to be executable when a new release of the package is built; invalid source code in the examples is not allowed. In addition to be available in HTML, the same help pages are available as a single LaTeX-compiled documentation for easy printing. At the moment of writing, this document consists of over 280 pages. See Figure 8 for an example. Additional help and examples are available online at `http://www.maths.lth.se/help/R/`.

5.3 Compatibility with other packages and software

Initially, this package was implemented as an object-oriented extension to the *sma* package [9], but is now a self-contained standalone package. The *RawData* and *RGData* classes are directly compatible with the structures in sma. In addition to this, classes that have virtual fields (see Section 6.4) that are named the same as the fields of the aforementioned two classes are compatible with sma. Similarly, converting sma data structures to aroma *MicroarrayData* objects is straightforward. See `?SMA` for more details. For this reason, the two packages can be used in parallel and all the functions of the sma package can be called using aroma objects. However, many of the functions in sma have been re-implemented in this package. The aroma package is compatible with methods provided by microarray packages such as *limma* [16] and other Bioconductor [17] packages. Moreover, for



GenePixData {aroma} R Documentation

The GenePixData class

Description

Package: aroma
Class GenePixData

```
Object
~~|
~~+--MicroarrayData
~~~~~~~|
~~~~~~~+--GenePixData
```

Directly known subclasses:

public abstract static class **GenePixData**
extends *MicroarrayData*

Usage

```
GenePixData(layout=NULL)
```

Arguments

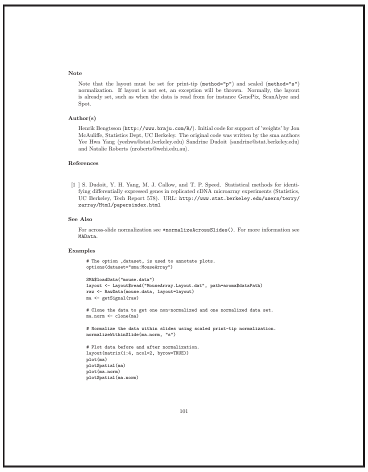
`layout` A `Layout` object specifying the spot layout of the slides in this data set.

Details

For information about the fields in this class see Axon's specifications of the GenePix File Formats at http://www.axon.com/GN_GenePix_File_Formats.html.

In addition to the above:

Flags



Note

Note that the layout must be set for print-tip (`method="p"`) and scaled (`method="s"`) normalization. If layout is not set, an exception will be thrown. Normally, the layout is already set, such as when the data is read from for instance GenePix, ScanAlyze and Spot.

Author(s)

Henrik Bengtsson (`http://www.braju.com/R/`). Initial code for support of 'weights' by Jon McAuliffe, Statistics Dept, UC Berkeley. The original code was written by the sma authors Yee Hwa Yang ⟨yeehwa@stat.berkeley.edu⟩ Sandrine Dudoit ⟨sandrine@stat.berkeley.edu⟩ and Natalie Roberts ⟨nroberts@wehi.edu.au⟩.

References

[1] S. Dudoit, Y. H. Yang, M. J. Callow, and T. P. Speed. Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments (Statistics, UC Berkeley, Tech Report 578). URL: `http://www.stat.berkeley.edu/users/terry/zarray/Html/papersindex.html`

See Also

For across-slide normalization see `*normalizeAcrossSlides()`. For more information see `MAData`.

Examples

```
# The option ,dataset, is used to annotate plots.
options(dataset="sma:MouseArray")

SMA$loadData("mouse.data")
layout <- Layout$read("MouseArray.Layout.dat", path=aroma$dataPath)
raw <- RawData(mouse.data, layout=layout)
ma <- getSignal(raw)

# Clone the data to get one non-normalized and one normalized data set.
ma.norm <- clone(ma)

# Normalize the data within slides using scaled print-tip normalization.
normalizeWithinSlide(ma.norm, "s")

# Plot data before and after normalization.
layout(matrix(1:4, ncol=2, byrow=TRUE))
plot(ma)
plotSpatial(ma)
plot(ma.norm)
plotSpatial(ma.norm)
```

101

Figure 8: Example of an HTML help page (left) and of a page in LaTeX-generated manual (right).

simple access to toolboxes exclusive to Matlab, we have implemented the *R.matlab* package [6] (in the *R.classes* bundle). It provides methods to connect to a local or a remote Matlab client, to transparently define Matlab variables and functions, and to make use of them from within R.

6 Philosophy of design and implementation

The package has been designed in a true object-oriented manner with an implementation making use of references variables. As discussed more below, passing objects by reference is useful if the objects are huge and if the objects are "modified" frequently. This is undoubtedly the case for microarray data and analysis. Moreover, references make it possible to create a more user-friendly methods interface, which is something that we have taken advantage of in this package. The rest of this section contains details that are not necessary to understand in order to start using aroma.

6.1 Underlying classes

In order to implement the above design, the *R.oo* package [4] has been developed. It is part of a package bundle named *R.classes* and provides the core functionality for object-oriented design with reference variables. R.oo is based exclusively on the S3/UseMethod programming paradigm [18], although future versions may in addition use S4/methods [11]. The package provides a transparent way of using reference variables for both the user and the developer. Passing objects to methods as pass-by-reference makes it possible for the methods to change the state (and the contents) of objects directly without having to return a modified copy. That is, objects are *mutable*. The standard approach in functional languages such as R is that all objects are non-mutable and a new instance is created whenever an object is modified. For further discussion of philosophical and practical considerations between the functional and object-oriented approaches, see [4]. Instances of classes that are derived from the root class *Object*, which is defined in R.oo, will automatically be accessed by reference variables.

The object-oriented core was separated from the rest of the microarray package and put into the R.oo package. The major advantage is that the R.oo package is likely to be used in other projects and by other developers and end users. A larger user base means more CPU test time, which increases the likelihood for identifying programming errors that will otherwise not be found by standard test procedures. Hence, a separate core package will improve the overall quality of the microarray package.

6.2 User interface

The aroma package does not have a GUI per se, but instead a API for access to all methods directly at R commando prompt. As shown by the examples in previous sections, reference variables and mutable objects make it possible to reduce the number of arguments that the user needs to specify. This way, the API can be kept simple. However, provided mutable objects, it should be straightforward to add a light-weighted third-party GUI utilizing, say, Tcl/Tk, which is part of the default R installation.

6.3 Programming style

In order to provide a well defined API simple enough to be used by anyone, *The R Coding Convention* (RCC) document [3] was developed. The RCC, which is an open suggestion to the R community, contains naming conventions similar to those found in Java, where classes, variables and methods can easily be differentiated by their names. For instance, names of classes are with mixed-case letters, starting with an upper-case letter, methods and variables start with a lower-case letter, and constants consists only of upper-case letters, e.g. `GenePixData`, `normalizeAffine()`, and `RED.HUE`. By enforcing the RCC, the time needed for a new user to become familiar with the API will be reduced.

6.4 Virtual fields

A *virtual field* is a field that is calculated *on demand* or *on the fly*. This can remove redundancy, which in turn will lower the risk for programming errors. It also makes the code more memory efficient. By providing a method, say, `getM()`, the virtual field `M` is automatically provided. Analogously, `setM(obj,value)` will add the option to assign values by `obj$M <- value`. Virtual fields are implemented by the *Object* class. With virtual fields, details can be hidden from the user, fields can be made read-only, and the correctness of assignment to fields can be asserted at the time of the assignment using specially written set methods. Moreover, with virtual fields it is possible to change the internal representation of the microarray objects and at the same time remain compatible with old packages, but also with new packages such as those developed under the Bioconductor project [17]. For instance, assume that a new normalization method that normalizes the fields `ch1` and `ch2` in a list structure is available in another package. For an aroma class to utilize this new method all that is needed is to provide the get and set methods `getCh1()` and `setCh1()`, respectively, and the same for `ch2`.

7 Obtaining the package

7.1 Requirements

In addition to a recent installation of R [13], the aroma package requires that the *R.classes* bundle is installed. The package can be used on a typical Windows, Mac OS X, various Linux systems, Sun Solaris and any operating system that supports R. Memory requirements are constrained by the size of the microarray data, but typically at least 256 Mb is required although 512-1024 Mb is recommended.

7.2 Installation and updates

To download and install aroma and its required packages, write

```
install.packages(c("R.classes", "aroma"),
                 contriburl="http://www.maths.lth.se/help/R")
```

For source code and manual installation go to `http://www.maths.lth.se/help/R/`. The R software is available from `http://www.r-project.org/`.

7.3 License

The R system is licensed under GNU General Public License (GPL). *R.classes* and aroma are licensed under GNU Lesser General Public License (LGPL). The GPL and LGPL can be obtained from `http://www.gnu.org/copyleft/` or by writing to the Free Software Foundation, Inc., 59 Temple Place – Suite 330, Boston, MA 02111-1307, USA.

Please cite this document whenever using aroma. When using *R.classes*, please cite [4].

References

[1] CSIRO Australia. *Spot user's manual.* CSIRO Mathematical and Information Sciences, Image Analysis Group, July 2003. http://spot.cmis.csiro.au/spot/.

[2] Henrik Bengtsson. Identification and normalization of plate effects in cDNA microarray data. Preprints in Mathematical Sciences 2002:28, Mathematical Statistics, Centre for Mathematical Sciences, Lund University, Sweden, 2002.

[3] Henrik Bengtsson. R Coding Conventions (draft), 2002. Published online: http://www.maths.lth.se/help/R/RCC/.

[4] Henrik Bengtsson. The R.oo package - object-oriented programming with references using standard R code. In Kurt Hornik, Friedrich Leisch, and Achim Zeileis, editors, *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*, Vienna, Austria, March 2003.

[5] Henrik Bengtsson. *Low-level analysis of microarray data*. PhD thesis, Centre for Mathematical Sciences, Division of Mathematical Statistics, Lund University, October 2004.

[6] Henrik Bengtsson. R.matlab - local and remote Matlab connectivity in R. Preprint in Mathematical Sciences (manuscript in progress), Mathematical Statistics, Centre for Mathematical Sciences, Lund University, Sweden, 2004. [manuscript in progress].

[7] Henrik Bengtsson and Ola Hössjer. Affine calibration for microarrays with dilution series or spike-ins. Preprints in Mathematical Sciences 2004:19, Mathematical Statistics, Centre for Mathematical Sciences, Lund University, Sweden, 2004.

[8] Henrik Bengtsson, Göran Jönsson, and Johan Vallon-Christersson. Calibration and assessment of channel-specific biases in microarray data with

extended dynamical range. Preprints in Mathematical Sciences 2003:37, Mathematical Statistics, Centre for Mathematical Sciences, Lund University, Sweden, 2003. Submitted.

[9] Ben Bolstad, Sandrine Dudoit, Ingrid Lönnstedt, Natalie Roberts, and Jean Yee Hwa Yang. R package: Statistics for Microarray Analysis (sma). http://www.stat.berkeley.edu/users/terry/zarray/Software/smacode.html, 2001.

[10] B.M. Bolstad, R.A. Irizarry, M. Åstrand, and T.P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–93, Jan 2003.

[11] John M. Chambers. *Programming with Data*. Springer, 1998.

[12] John M. Chambers and Duncan Temple Lang. Object-oriented programming in R. *R News*, 1(3):17–19, September 2001.

[13] Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

[14] Friedrich Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In W. Härdle and B. Rönz, editors, *Compstat 2002 - Proceedings in Computational Statistics*, pages 575–580. Physika Verlag, Heidelberg, Germany, 2002.

[15] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2003.

[16] Gordon Smyth and Terry Speed. Normalization of cDNA microarray data. *METHODS: Selecting Candidate Genes from DNA Array Screens: Application to Neuroscience (to appear)*, April 2003.

[17] Bioconductor Core Team. Bioconductor - open source software for bioinformatics. http://www.bioconductor.org/, 2002.

[18] W.N. Venables and B.D. Ripley. *Modern Applied Statistics with S-PLUS*. Springer, third edition, 1999.

[19] Yee Hwa Yang, Michael Buckley, Sandrine Dudoit, and Terry Speed. Comparison of methods for image analysis on cDNA microarray data. Technical Report 584, Department of Statistics, University of California at Berkeley, Nov 2000.

[20] Yee Hwa Yang, Sandrine Dudoit, Percy Luu, and Terence P. Speed. Normalization for cDNA microarray data. In Michael L. Bittner, Yidong Chen, Andreas N. Dorsel, and Edward R. Dougherty, editors, *Proceedings of SPiE*, volume 4266 of *Microarrays: Optical Technologies and Informatics*, pages 141–152, San Jose, California, June 2001. The International Society for Optical Engineering.

G

Paper G

Affine calibration for microarrays with dilution series or spike-ins

Henrik Bengtsson[1] and Ola Hössjer[2]

[1] Mathematical Statistics at the Centre for Mathematical Sciences, Lund University, Box 118, SE-221 00 Lund, Sweden. Email: hb@maths.lth.se.

[2] Mathematical Statistics at the Mathematics Department, Stockholm University.

Abstract

Background: In this theoretical study we follow up our previous theoretical and applied work on affine calibration and normalization methods by suggesting a stochastic affine model for two- or multi-channel microarrays containing dilution series. With replicated probes (spots) at various, not necessarily known, concentration levels it is possible to identify the channel biases uniquely. For this to be true it is necessary that some genes are differentially expressed, but which they are or what their relative gene-expression levels are do not have to be known. The method presented may also be applied to so called spike-in data. From the maximum likelihood estimator, estimates of the relative gene-expression levels for all dilution series follow directly. Given the estimated channel biases and scale factors, back transformation gives calibrated probe signals that are proportional to the amount of fluorophore in each channel. With the assumption that most genes are non-differentially expressed, the proposed calibration method makes the signals in each channel proportional to the corresponding gene-expression levels.

Results: The model suggested is a heteroscedastic affine model where the standard deviations of the error terms are equal for all replicates that belong to the same gene, but may vary freely between genes. For a specific probe, the standard deviation in one channel is assumed to be proportional to the other channels. These constraints were chosen as a compromise between model flexibility and feasible parameter estimates. We investigate the properties of the parameter estimates from simulated data by comparing bootstrap and asymptotic confidence intervals based on normal approximation for different setups of number of genes and replicates. We simulate two-channel gene-expression data with technical dilution series from the suggested model and, in order to investigate the robustness of

the estimator against model misspecification, a more general model for which the standard deviations are not only proportional to the gene-expression levels, but also the probe concentrations. We find that the standard deviation of the bias-parameter estimates is inversely proportional to the square root of the product of the number genes and replicates, provided the latter is not too small. For modest number of replicates, the standard deviation based on the Fisher information is a good approximation to the bootstrap ditto. When too few replicates are available, normality can not be assumed and the standard errors are under-estimated. For similar reasons, for different setups of number of genes and replicates such that the total number of probes are the same, we found that it is better to use more replicates and fewer genes, if possible. Comparing dilution series with uniformly and logarithmic-uniformly distributed concentrations better results are obtained using the latter.
Availability: All methods have been implemented in the cross-platform R package called aroma, which is available for free from `http://www.braju.com/R/`.

Keywords: spotted microarrays; systematic effects; bias; calibration; normalization; dilution series; technical replicates; profile likelihood; Fisher information; stochastic affine model; heteroscedastic noise..

1 Introduction

On a microarray there is a large number of disjoint *probes*, which each consists of a homogeneous set of immobilized DNA with a known or at least an identifiable sequence. For a typical microarray experiment, for each cell sample for which a gene-expression profile is to be obtained, RNA is extracted and reversely transcribed into cDNA. In the same or a following process, the cDNA is labeled with a unique dye, commonly a fluorescent dye. The set of cDNA is referred to as the *target*. In single-channel technique identically labeled cDNA constitutes the target whereas in two-channel techniques two differently labeled cDNA sets of equal amount constitute the target. The target is then let to hybridize to the microarray so that, ideally, sequences of target cDNA that match the DNA in a specific probe will hybridize to that probe and nowhere else. The effect is a massive parallel and spatial sort of target sequences to the corresponding probes. Next, using a microarray scanner the microarray is scanned in different wavelength channels, one for each dye used, resulting in one or several high-resolution spatial images. Using image analysis techniques, the pixels in the images corresponding to the probe regions are identified and the amount of labeled cDNA in each probe and channel is quantified based on the pixel intensities.

Thus, the main objective when *scanning a microarray* is to quantify the amount of fluorescent molecules in each probe and each channel. Then, by assuming that the amount of hybridized DNA is proportional to the amount of fluorophores and so on, estimates of the gene-expressions can be obtained. However, in this paper we will focus solely on the problem of correctly quantifying the amount of different fluorophores in the probes.

As an extension to our previous work [3, 2], we suggest an affine *calibration (normalization) method* based on a stochastic model for two- or multi-channel microarrays containing *dilution series*. With minor additional assumptions, the same calibration method can used for so called *spike-in calibration data* where genes with known amounts of cDNA have been added. Without being further discussed, with additional assumptions it may also be applied to gene-expression signals obtained from multiple single-channel microarrays.

The outline of the paper is as follows. In the “Model” section, we present the stochastic affine model describing scanned microarray data with within-array replicates. We outline the necessary assumptions and model constraints in order to make all parameters identifiable. In the following “Methods” section, we provide a model estimator based on maximum likelihood techniques, which forms the basis for the calibration method described thereafter. Different properties of the parameter estimates are discussed as well as the two bootstrap methods used. How the microarray data is simulated from two different models and with two types of dilutions series is described in the “Data” section. In the “Results” section, the properties of the estimates for microarrays with different number of genes, different number of replicates, and different types of replicates etc. are presented. A brief discussion then follows and conclusions are given in the last section.

2 Model

Consider a microarray experiment involving genes $i = 1, \ldots, I$ from RNA extracts $c = 1, \ldots, C$ with $C \geq 2$. From now on, we will refer to each set of signals from each RNA extract as a *channel*. Let $\mu_{c,i}$ be the true gene-expression (transcription) level of gene i in channel c. Ideally, statistical analysis is done on these quantities, but here we will focus on the problem of quantifying the (relative) amount of fluorescent dyes of a certain kind in each probe.

Within-array technical replicates are probes on the microarray that have identical nucleotide sequences. For instance, in spotted microarrays, technical replicates are spots that are printed with the same probe sequence from the same clone library. Formally we say that for gene i there exist replicates $j = 1, \ldots, J_i$ on the microarray so that in total there are $J_\Sigma = \sum_{i=1}^{I} J_i$ probes on the array. Let index (i, j) refer to the j^{th} replicated probe for gene i. Furthermore, let $z_{c,i,j}$ be the *concentration of fluorescent molecules* that are attached to DNA sequences *hybridized* to probe (i, j) in channel c. Note, some of these may sit on cross-hybridized targets. However, we do not differentiate between cross-hybridized and true target DNA in this study. With the assumption that the relative amount of true and cross-hybridized target DNA is the same for all replicated probes of the same gene, we have, if assuming everything else is equal, that

$$z_{c,i,j} = z_{c,i};\ \forall j. \tag{1}$$

The wavelength profiles of the fluorescent light from replicated probes are expected to be equal up to a global scale factor.

A *dilution series* is a set of within-array technical replicates where the replicated probes have different concentrations of DNA sequences. Let the *probe concentration* of probe (i, j) be denoted by $b_{i,j}$ (in arbitrary units). Moreover, assume that the brightness (excitation coefficient times quantum yield) of a probe is proportional to the probe concentration and the amount of excitation light, which in turn is assumed to be constant, and can be summarized in a channel-specific quantity called b_c. The total amount of fluorescent light $x_{c,i,j}$ from probe (i, j) with “gain” b_c can then be modeled as

$$b_c x_{c,i,j} = b_c b_{i,j} z_{c,i};\ \forall c. \tag{2}$$

We assume no spectral cross-talk and no saturation. Added to the fluorescent light is also stray white and stray laser light, auto-fluorescent light, photocathode noise, electronic noise [3] etc., which we assume all can be modeled as a channel-specific but probe-independent constant a_c plus zero-mean noise.

2.1 The affine stochastic model

Let $y_{c,i,j}$ be the observed expression-level for replicate j of gene i in channel c. We assume that

$$\begin{aligned} y_{c,i,j} &= a_c + b_c x_{c,i,j} + \varepsilon_{c,i,j} \\ &= a_c + b_c b_{i,j} z_{c,i} + \varepsilon_{c,i,j};\ \forall c, i, j \end{aligned} \tag{3}$$

where $\varepsilon_{c,i,j}$ is independent noise with $E[\varepsilon_{c,i,j}] = 0$ and $V[\varepsilon_{c,i,j}] = \sigma^2_{c,i,j}$. Furthermore, assume that

$$\sigma_{c,i,j} = \sigma_{c,i} \tag{4}$$

$$\sigma_{c,i} = k_c \sigma_i \tag{5}$$

for $c = 1, \ldots, C$ where $k_1 = 1$. In words, this means that the relative standard deviation of the noise in the different channels does not dependent on replicate. Let

$$\beta_{c,i} = \frac{b_c z_{c,i}}{b_1 z_{1,i}} = \frac{b_c x_{c,i,j}}{b_1 x_{1,i,j}}; \forall j. \tag{6}$$

Thus, $\beta_{c,i}$ is up to a scale factor equal to $z_{c,i}/z_{1,i}$, which is the “expression level” of gene i in channel c in units of the corresponding expression level in channel 1. With the above parametrization and no further model assumptions b_c and $x_{c,i,j}$ are non-identifiable. Define $\gamma_{c,i,j} = b_c x_{c,i,j}$. The model can then be written as

$$y_{c,i,j} = a_c + \beta_{c,i}\gamma_{1,i,j} + \varepsilon_{c,i,j}; \forall c, i, j \tag{7}$$

where all parameters are identifiable. The $2C - 1$ structural parameters, here the parameters independent of gene, are $\boldsymbol{\theta} = (\mathbf{a}, \mathbf{k})$ where $\mathbf{a} = (a_1, \ldots, a_C)$ and $\mathbf{k} = (1, k_2, \ldots, k_C)$. The $C + J_i$ incidental parameters for gene i are $\boldsymbol{\xi}_i = (\boldsymbol{\beta}_i, \boldsymbol{\gamma}_{1,i}, \sigma_i)$, where $\boldsymbol{\beta}_i = (1, \beta_{2,i}, \ldots, \beta_{C,i})$ and $\boldsymbol{\gamma}_{1,i} = (\gamma_{1,i,1}, \ldots, \gamma_{1,i,J_i})$. Let $\boldsymbol{\xi} = \{\boldsymbol{\xi}_i\}_i$, $\boldsymbol{\beta} = \{\boldsymbol{\beta}_i\}_i$, $\boldsymbol{\gamma}_1 = \{\boldsymbol{\gamma}_{1,i}\}_i$, and $\boldsymbol{\sigma} = \{\sigma_i\}_i$. In total there are $|\boldsymbol{\theta}| + |\boldsymbol{\xi}| = C(I + 2) + J_\Sigma - 1 = p$ parameters.

3 Methods

We will next present how the above model with Gaussian noise can be estimated using maximum likehood techniques.

3.1 Maximum likelihood estimation

Let $\mathbf{y} = \{\mathbf{y}_i\}_i$ where $\mathbf{y}_i = (y_{1,i,1}, \ldots, y_{1,i,J_i}, \ldots, y_{C,i,1}, \ldots, y_{C,i,J_i}) \in \mathbb{R}^{CJ_i}$ is the vector of the observations for gene i in all channels and all replicates $j = 1, \ldots, J_i$. The log-likelihood function of the parameters given the data is then

$$l(\boldsymbol{\theta}, \boldsymbol{\xi}; \mathbf{y}) = \sum_{i=1}^{I} l_i(\boldsymbol{\theta}, \boldsymbol{\xi}_i; \mathbf{y}_i). \tag{8}$$

Assuming Gaussian zero-mean noise with probe specific variation

$$\varepsilon_{c,i,j} \in N(0, \sigma^2_{c,i,j}), \tag{9}$$

the log-likelihood for observation $y_{c,i,j}$ is up to an additive constant

$$l_{c,i,j}(\boldsymbol{\theta}, \boldsymbol{\xi}_i; y_{c,i,j}) = -\ln \sigma_{c,i,j} - \frac{1}{2}\left(\frac{y_{c,i,j} - (a_c + \gamma_{c,i,j})}{\sigma_{c,i,j}}\right)^2 \tag{10}$$

With assumptions (4) and (5), the log-likelihood for gene i becomes

$$\begin{aligned} l_i(\boldsymbol{\theta}, \boldsymbol{\xi}_i; \mathbf{y}_i) &= \sum_{c=1}^{C} \sum_{j=1}^{J_i} l_{c,i,j}(\boldsymbol{\theta}, \boldsymbol{\xi}_i; y_{c,i,j}) \\ &\overset{(4,5)}{=} -\sum_{c=1}^{C} \sum_{j=1}^{J_i} \ln(k_c \sigma_i) - \frac{1}{2} \sum_{c=1}^{C} \sum_{j=1}^{J_i} \left(\frac{y_{c,i,j} - a_c - \gamma_{c,i,j}}{k_c \sigma_i}\right)^2 \\ &= -J_i \sum_{c=2}^{C} \ln k_c - C J_i \ln \sigma_i - \frac{1}{2\sigma_i^2} \sum_{c=1}^{C} \frac{1}{k_c^2} \sum_{j=1}^{J_i} (y_{c,i,j} - a_c - \gamma_{c,i,j})^2. \end{aligned} \tag{11}$$

Finally, the overall log-likelihood is

$$\begin{aligned} l(\boldsymbol{\theta}, \boldsymbol{\xi}; \mathbf{y}) = {} & -J_\Sigma \sum_{c=2}^{C} \ln k_c - C \sum_{i=1}^{I} J_i \ln \sigma_i \\ & - \frac{1}{2} \sum_{c=1}^{C} \frac{1}{k_c^2} \sum_{i=1}^{I} \frac{1}{\sigma_i^2} \sum_{j=1}^{J_i} (y_{c,i,j} - a_c - \gamma_{c,i,j})^2. \end{aligned} \tag{12}$$

3.1.1 Estimation of structural parameters

The profile log-likelihood for the structural parameters is

$$l_p(\boldsymbol{\theta};\mathbf{y}) = \sup_{\boldsymbol{\xi}} l(\boldsymbol{\theta},\boldsymbol{\xi};\mathbf{y}) = \sum_{i=1}^{I} \sup_{\boldsymbol{\xi}_i} l_i(\boldsymbol{\theta},\boldsymbol{\xi}_i;\mathbf{y}_i) = \sum_{i=1}^{I} l_i(\boldsymbol{\theta},\hat{\boldsymbol{\xi}}_i(\boldsymbol{\theta});\mathbf{y}_i) \quad (13)$$

where $\hat{\boldsymbol{\xi}}_i(\boldsymbol{\theta})$ maximizes (11) given $\boldsymbol{\theta}$. Finally,

$$\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} l_p(\boldsymbol{\theta};\mathbf{y}) \quad (14)$$

is the maximum likelihood estimate of $\boldsymbol{\theta}$. In practice, we search over an iteratively refined $(2C-1)$-dimensional grid box constrained for $\boldsymbol{\theta}$.

3.1.2 Estimation of incidental parameters

The parameter vector $\boldsymbol{\xi}_i = (\boldsymbol{\beta}_i, \gamma_{1,i}, \sigma_i)$ can be estimated as follows. We begin by estimating the parameters $\boldsymbol{\beta}_i = (1, \beta_{2,i}, \ldots, \beta_{C,i})$ given the assumption that $\boldsymbol{\theta}$ is the true structural parameter. Let

$$\begin{aligned}\tilde{y}_{c,i,j} &= \frac{y_{c,i,j} - a_c}{k_c} = \frac{\gamma_{c,i,j} + \varepsilon_{c,i,j}}{k_c} = \frac{\beta_{c,i}\gamma_{1,i,j} + \varepsilon_{c,i,j}}{k_c} \\ &= \tilde{\beta}_{c,i}\gamma_{1,i,j} + \tilde{\varepsilon}_{c,i,j}\end{aligned} \quad (15)$$

where $\tilde{\beta}_{c,i} = \beta_{c,i}/k_c$ and $\tilde{\varepsilon}_{c,i,j} = \varepsilon_{c,i,j}/k_c$ so that $V[\tilde{y}_{c,i,j}] = V[\tilde{\varepsilon}_{c,i,j}] = \sigma_i^2$. Thus, given the true structural parameters $\boldsymbol{\theta}$ with noise-free observations, the J_i translated and rescaled points $\tilde{\mathbf{y}}_{i,j} = (\tilde{y}_{1,i,j}, \ldots, \tilde{y}_{C,i,j}), j = 1, \ldots, J_i$ in $\mathbb{R}^C$ are located along the line $\tilde{L}_i(\mathbf{0}, \tilde{\boldsymbol{\beta}}_i)$ that goes through the origin and has direction $\tilde{\boldsymbol{\beta}}_i = (\tilde{\beta}_{1,i}, \ldots, \tilde{\beta}_{C,i})$ with $\tilde{\beta}_{c,i} \geq 0; \forall c$. With independent and identically distributed noise, the observations cluster along the line $L(\mathbf{0}, \tilde{\boldsymbol{\beta}}_i)$ and estimates of $\tilde{\beta}_{c,i}$ and $\gamma_{1,i,j}$ given $\boldsymbol{\theta}$ are found from the line in $\mathbb{R}^C$ that minimizes the sum of squared orthogonal distances to $\{\tilde{\mathbf{y}}_{i,j}\}_j$. A version of principal component analysis (PCA) where data is not centered in advance can be used to estimate this linear subspace, where the direction of the estimated $\tilde{L}_i$ is defined by the first principal component of $\tilde{\mathbf{y}}_i = \{\tilde{\mathbf{y}}_{i,j}\}_j$, which is described by the components of $\mathbf{v}_i = (v_{1,i}, \ldots, v_{C,i})$; $\|\mathbf{v}_i\| = 1$. Each component is then, up to a scale factor, an estimate of the corresponding $\tilde{\beta}_{c,i}$. Since $\beta_{1,i} = k_1 = 1$, we obtain

$$\hat{\beta}_{c,i}(\boldsymbol{\theta}) = \frac{k_c}{v_{1,i}} \cdot v_{c,i};\ \forall c. \quad (16)$$

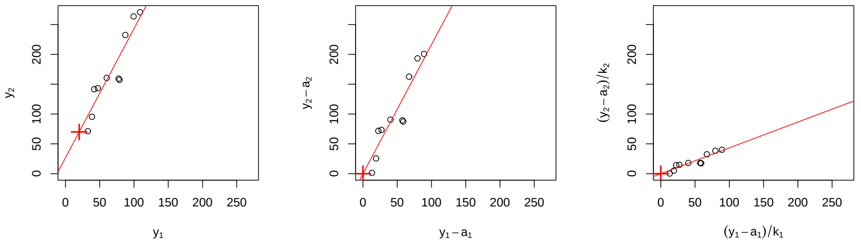


Figure 1: Example of the estimate of $\boldsymbol{\beta}_i$ and $\boldsymbol{\gamma}_{1,i}$ for $C = 2$ and $J_i = 10$. *Left*: J_i data points in (y_1, y_2). $\mathbf{a} = (20, 70)$ is marked with a cross. *Middle*: The same data points in the translated space. *Right*: The translated and rescaled ($\mathbf{k} = (1, 5)$) space in which the PCA fit is done. The fit is displayed as a line. The same fit is illustrated in the other spaces too by back transformation.

To estimate $\boldsymbol{\gamma}_{1,i}$, we observe that when projecting the j^{th} replicate $\tilde{\mathbf{y}}_{i,j}$ of gene i onto the PCA estimated line $\tilde{L}_i(\mathbf{0}, \mathbf{v}_i)$, the first coordinate of the projection is $\hat{\gamma}_{1,i,j}(\boldsymbol{\theta})$ and its distance from the origin is $\lambda_{i,j}$. From this it follows

$$\hat{\gamma}_{1,i,j}(\boldsymbol{\theta}) = v_{1,i}\lambda_{i,j};\ \forall j. \tag{17}$$

Finally, given $\boldsymbol{\theta}$ and estimates of $\beta_{c,i}$ and $\{\gamma_{1,i,j}\}_j$, a bias-corrected estimate of σ_i^2 is

$$\hat{\sigma}^2_{\text{corr},i}(\boldsymbol{\theta}) = \frac{1}{J_iC - (C + J_i - 1)} \cdot \sum_{c=1}^{C} \frac{1}{k_c^2} \sum_{j=1}^{J_i} (y_{c,i,j} - a_c - \hat{\beta}_{c,i}(\boldsymbol{\theta})\hat{\gamma}_{1,i,j}(\boldsymbol{\theta}))^2. \tag{18}$$

Thus, given $\boldsymbol{\theta} = (\mathbf{a}, \mathbf{k})$ we have obtained an estimate of $\boldsymbol{\xi}_i = (\boldsymbol{\beta}_i, \boldsymbol{\gamma}_{1,i}, \sigma_i)$ for gene i.

3.1.3 Calibration

With the assumption that most probes are non-differentially expressed, it is straightforward to estimate $b_c; \forall c$. Hence, estimates of the channel c to channel one relative concentration of fluorescent molecules $\{z_{c,i}/z_{1,i}\}_{c,i}$ follow immediately from (6) and the maximum likelihood estimates of $\{\beta_{c,i}\}_{c,i}$. If it is only these quantities, which each summarizes all signals from a complete calibration series, that are of interest, no further analysis or pre-processing of the signals is needed. However, if for various reasons further calibration or normalization is to be applied on the probes separately, or another method to combine the signals from the replicated probes is to be used, the calibrated probe signals can be calculated as

$$\hat{x}_{c,i,j} = \frac{y_{c,i,j} - \hat{a}_c}{\hat{b}_c};\ \forall c, i, j. \tag{19}$$

3.2 Properties of the estimates

Main focus will be on the properties of the bias-parameter estimates $\{a_c\}_c$ as these are of major importance in order to calibrate the measurements correctly, but we will also look at the estimates of the relative gene-expression levels $\{\beta_{c,i}\}_{c,i}$. Especially, we study the standard deviation and confidence intervals of the estimates. Such can be obtained both by large sample normal approximations and from bootstrap simulations, as described next.

3.2.1 Confidence interval from Fisher information

From model (3) with error structure (5), it is possible to obtain a closed form expression for the Fisher information of the bias-parameter estimates and, hence, standard deviation and confidence interval estimates. For large samples, classical theory gives that the parameter estimates are approximately normally distributed, with a standard deviation that can be estimated as the square-root of the diagonal terms of the inverse of the $p \times p$ Fisher information matrix $\mathcal{I}$. In particular, for the bias parameters $a_1, \ldots, a_C$ we obtain $\hat{\sigma}^2_{a_c} = \hat{V}[\hat{a}_c] \approx (\hat{\mathcal{I}}^{-1})_{a_c a_c}$ where $\hat{\mathcal{I}}$ is an estimate of $\mathcal{I}$ obtained by replacing unknown parameter values by estimates. As shown in appendix A, for $C = 2$ we have

$$1/(\mathcal{I}^{-1})_{a_c a_c} = \frac{1}{\bar{\beta}^2_c} \sum_{i=1}^{I} w_i (\beta_{2,i} - \bar{\beta}_2)^2 \tag{20}$$

where $\bar{\beta}_c = \left(\sum_{i=1}^I w_i \beta_{c,i}\right) / \sum_{i=1}^I w_i$ is a weighted average of all "relative expression levels" in channel c, and similar for $\bar{\beta^2_c} = \left(\sum_{i=1}^I w_i \beta^2_{c,i}\right) / \sum_{i=1}^I w_i$.

The weights are

$$w_i = \frac{J_i}{\sigma_i^2(k_2^2 + \beta_{2,i}^2)} \cdot \frac{\sum_{j=1}^{J_i}(b_{i,j} - \bar{b}_i)^2}{\sum_{j=1}^{J_i} b_{i,j}^2} \tag{21}$$

where $\bar{b}_i = \sum_{j=1}^{J_i} b_{i,j}/J_i$ is the average probe concentration for gene i. It is interesting to notice from (20)-(21) that the asymptotic standard error $\hat{\sigma}_{a_c}$ decreases the more the expression levels $\{\gamma_{1,i,j}\}_j$ vary between replicates. Equivalently, the more similar the $\{b_{i,j}\}_j$ are, the smaller the Fisher information is. Moreover, when the relative expression levels $\{\beta_{2,i}\}_i$ are close to each other the standard error is large.

It now follows that an approximate $(1-\alpha)$-level confidence interval for bias parameter a_c is

$$R_{a_c} = \left[\hat{a}_c - \lambda_{\alpha/2}\hat{\sigma}_{a_c}, \hat{a}_c + \lambda_{\alpha/2}\hat{\sigma}_{a_c}\right] \tag{22}$$

where $\lambda_\alpha = \Phi^{-1}(1-\alpha)$ and $\Phi(\cdot)$ is the distribution function for $N(0,1)$. Note that, with simulated data we can obtain standard deviations and confidence intervals using both the true incidental parameters $\{k_c\}_c$, $\{\sigma_i\}_i$, $\{\beta_{2,i}\}_i$, and $\{\gamma_{1,i,j}\}_{i,j}$, as well as their estimates.

For the gene-expression levels we have $\hat{\sigma}^2_{\beta_{c,i}} = \hat{V}(\hat{\beta}_{c,i}) \approx (\hat{\mathcal{I}}^{-1})_{\beta_{c,i},\beta_{c,i}}$. It is shown in the appendix that for $C = 2$,

$$1/(\mathcal{I}^{-1})_{\beta_{2,i},\beta_{2,i}} = \frac{z_{1,i}^2 \sum_{j=1}^{J_i} b_{i,j}^2}{\sigma_i^2(k_2^2 + \beta_{2,i}^2)} + O(J_\Sigma^{-1}), \tag{23}$$

where the remainder term of size $O(J_\Sigma^{-1})$ is present since the bias parameters a_1 and a_2 are unknown and have to be estimated. For large J_Σ we may drop this term and obtain confidence intervals $R_{\beta_{2,i}}$ analogously to (22).

3.2.2 Confidence interval from bootstrap samples

Alternatively, standard deviation and confidence interval may be calculated from B bootstrap estimates obtained from both parametric and nonparametric bootstrap methods [5], as described next. Let $\boldsymbol{\gamma}_i = (\gamma_{1,i,1}, \ldots, \gamma_{1,i,J_i}, \ldots, \gamma_{C,i,1}, \ldots, \gamma_{C,i,J_i}) \in \mathbb{R}^{CJ_i}$. Moreover, let $\boldsymbol{\mu}_i = (a_1\mathbf{1}_{J_i}, \ldots, a_C\mathbf{1}_{J_i}) + \boldsymbol{\gamma}_i$, and let

$\mathbf{\Sigma}_i = \sigma_i^2 \cdot \text{diag}(\mathbf{1}_{J_i}, k_2^2\mathbf{1}_{J_i}, \ldots, k_C^2\mathbf{1}_{J_i})$ be a diagonal $(CJ_i)\times(CJ_i)$ matrix where $\mathbf{1}_n$ is a vector of ones of length n. Now, with independent Gaussian noise (9), model (7) can be written as

$$\mathbf{y}_i \in N\left(\boldsymbol{\mu}_i, \mathbf{\Sigma}_i\right);\ \forall i. \tag{24}$$

First, we consider the *parametric* bootstrap method where the gene vectors $\boldsymbol{\mu}_i$ and the error terms $\{\varepsilon_{c,i,j}\}_{c,j}$ are resampled using the *estimated* parameters. Let $\hat{\boldsymbol{\mu}}_i$ and $\hat{\mathbf{\Sigma}}_i$ be defined in analogue to the above but with estimates $\{\hat{a}_c\}_c$, $\hat{\boldsymbol{\gamma}}_i$, $\hat{\sigma}_i$, and $\{\hat{k}_c\}_c$ replacing the true parameters. A parametric bootstrap method is then

$$\mathbf{y}_i^* \in N\left(\hat{\boldsymbol{\mu}}_i, \hat{\mathbf{\Sigma}}_i\right);\ \forall i. \tag{25}$$

With this approach, it is possible to obtain confidence intervals for the structural as well as the gene-specific incidental parameters. Second, we consider the *non-parametric* method where we *resample from genes*;

$$\mathbf{y}_i^* = \mathbf{y}_{u_i};\ \forall i \tag{26}$$

where $\{u_i\}_i$ are independent and uniformly distributed indices on $i = 1\ldots, I$. Because J_i is typically small, it is *not* reasonable to bootstrap within genes this way, that is to *resample from replicates*. For instance, with $J_i = 2$ half of the samples would contain genes with identical replicated signals making it impossible to estimate $\beta_{c,i}$ and $\{\gamma_{1,i,j}\}_j$. The nonparametric method (26) is somewhat simpler to implement, especially in a distributed computational environment since it does not require parameter estimates in order to resample. However, contrary to the parametric approach, it only allows us to calculate confidence intervals for the structural parameters.

Continuing, we estimate the standard deviation both by the sample standard deviation, but also by the median absolute deviation (MAD) of the bootstrap estimates. An α-level percentile confidence interval $\tilde{R}_{a_c}$ for bias parameter a_c in channel c is calculated from the bootstrap estimates $\{\hat{a}_c^*\}$ as

$$\tilde{R}_{a_c} = (\hat{a}^*_{c;(i_1)}, \hat{a}^*_{c;(i_2)}) \tag{27}$$

where $\hat{a}^*_{c;(i)}$ is the i^{th} ranked $\hat{a}_c^*$, $i_1 = [B\alpha/2]$ and $i_2 = [B(1-\alpha/2)]$ with $\alpha \in [0,1]$. A *bias-corrected* version of this is where $i_1 = [B\rho(\alpha/2)]$

and $i_2 = [B\rho(1-\alpha/2)]$ with quantiles $\rho(\alpha) = \Phi(2\lambda + \lambda_{1-\alpha})$ and $\lambda = \Phi^{-1}(P^*(\hat{a}_c^* \leq \hat{a}_c))$. We estimate $P^*(\hat{a}_c^* \leq \hat{a}_c)$ by the proportion of bootstrap estimates $\hat{a}_c^*$ that are less or equal to the original estimate $\hat{a}_c$.

Similar to the above, we can calculate a confidence interval $\tilde{R}_{\beta_{c,i}}$ for each (relative) gene-expression level $\beta_{c,i}; c = 2, \ldots, C; i = 1, \ldots, I$[17]. The confidence interval for gene i in channel c from bootstrap estimates is

$$\tilde{R}_{\beta_{c,i}} = (\hat{\beta}^*_{c,i;(i_1)}, \hat{\beta}^*_{c,i;(i_2)}) \tag{28}$$

with notations analogue to the above. Percentile or bias-corrected estimates can be obtain by choosing i_1 and i_2 as before.

3.2.3 Average gene-expression coverage

Given confidence-interval estimates $\{\tilde{R}_{\beta_{c,i}}\}_i$ and the true $\{\beta_{c,i}\}_i$ parameters, we calculate the *average coverage* of the relative gene-expression levels in channel c as

$$\hat{C}_{\beta_c} = \frac{1}{I}\sum_{i=1}^{I} \mathbb{I}(\beta_{c,i} \in \tilde{R}_{\beta_{c,i}}) \tag{29}$$

where $\mathbb{I}(\cdot)$ is the indicator function. Note, to calculate the degree of coverage $\hat{C}_{\beta_{c,i}}$ for each individual gene, and similarly for a_c, another, say, $T-1$ sets of samples, which each has B bootstrap estimates, have to be generated. Due to time constraints, this was not feasible.

4 Data

All data in this report is based on simulated data as we wish to study the properties of the above estimates for the different model and parameter setups. Real-data experiments are planned, but not covered in this theoretical study.

4.1 Simulated models

We simulate microarray data from the affine model (3) with independent error terms $\varepsilon_{c,i,j} \in N(0, \sigma^2_{c,i,j})$ and, in order to investigate the estimator's robustness

[17] Since $\beta_{1,i} = 1$ by definition, there is no confidence interval for $c = 1$.

toward model misspecification, with two different error structures:

$$\sigma_{c,i,j} = \bar{z}_i k_c \sigma \tag{30}$$

$$\sigma_{c,i,j} = b_{i,j} b_c z_{c,i} k_c \sigma \tag{31}$$

for all channels, genes and replicates where $\bar{z}_i = \sum_{c=1}^{C} z_{c,i}$, b_c, $b_{i,j}$, and $z_{c,i}$ are as in (3) with a proportionality constant $k_c\sigma$ that depends on the channel c and satisfies the constraint $k_1 = 1$. Note that the former error structure (30) is a special case of (4)-(5) such that the standard deviation of the noise is proportional to the amount of DNA hybridized to the dilution series as a whole, cf. (1). The latter error structure is such the standard deviation is proportional to the probe concentration as well as the channel-specific amount of hybridization, cf. [13, 12, 6].

For both error structures we study two different types of dilution series, one with uniformly and one with log-uniformly distributed concentrations:

$$b_{i,j} = j/J_i \tag{32}$$

$$b_{i,j} = 2^{(j-J_i)/J_i} \tag{33}$$

such that $0 < b_{i,j} \leq 1$ and $b_{i,J_i} = 1$ for all genes.

Without loss of generality, we have chosen to study the above for setups with two channels, the same number of replicates for all genes, the same biases, the same scale parameters, and the same relative standard deviation for all channels. More precisely, $C = 2$, $J_i = J; \forall i$, $\mathbf{a} = (20, 70)$, $b_c = 1$, and $k_c = 1; \forall c$. For error structure (30) we use $\sigma = 1/16$ and for (31) we use $\sigma = 1/4$.

4.2 Simulation of “expressions level”

The expression levels $\{z_{c,i}\}_{c,i}$ are simulated as follows. First we sample from $\log_2 \gamma_{1,i} \in \Gamma(1, 1/3)$ with shape parameter 1 and rate parameter $1/3$. Samples for which $\log_2 \gamma_{1,i} > 16$ are censored to 16. To simulate fold changes we sample from $\log_2 \beta_{c,i} \in N(0, 2^2)$ and assign $z_{c,i} \leftarrow \beta_{c,i}\gamma_{1,i} + \delta; \forall c$ where $\delta = 100$ is a small constant in order to avoid gene expressions close to zero.

4.3 Simulated array setups

For the nonparametric bootstrap method, we have simulated data from the model setups and obtained original and $B = 500$ bootstrap estimates of all structural and incidental parameters for most combinations of $I = 100, 300, 600, 1000, 1500$ and $J = 2, 3, 4, 5, 10, 30, 100$. For the parametric bootstrap method, we choose slightly different sets of I and J, but of the same order.

5 Results

With the data sets in hand, we have studied how the standard deviation of the bias-parameter estimates vary i) with varying number of replicates J and fixed number of genes I, ii) with varying I and fixed J, and iii) with different combinations of I and J such that the total number of probes, $J_\Sigma = IJ$, is fixed. We also compare the estimates obtained from arrays with $\log_2$-uniformly and uniformly distributed dilution series. Similarly, we investigate the properties of the relative expression estimates $\{\beta_{c,i}\}_{c,i}$ and the average coverage of these. We end this section with a visual comparison of non-calibrated and calibrated *probe* signals, as well as a comparison between estimated and true relative *gene* expression ratios. All analysis was done with data simulated under the correct model (30) and the misspecified model (31).

5.1 Varying number of replicates

5.1.1 Correct model

Bias-parameter estimates: For a constant number of genes, the standard deviation of the bias-parameter estimates decreases as shown in the left graphs of Figure 2. For not too small number of replicates (or genes), the standard deviation estimates based on the *estimated* Fisher information, which is fast to calculate, are very close to the estimates based on computationally intensive bootstrap simulations. However, for a typical microarray experiment where the number of replicates is small (here about $J < 5$ or so depending on I), the former tend to be too small. See also Figure 5. We believe this is because the asymptotic normality assumption does not hold for small J as illustrated by the density plot in the middle graph at the top. Moreover, for large number of genes, the standard deviation calculated from the *true* Fisher information is close the estimated ditto. For small number

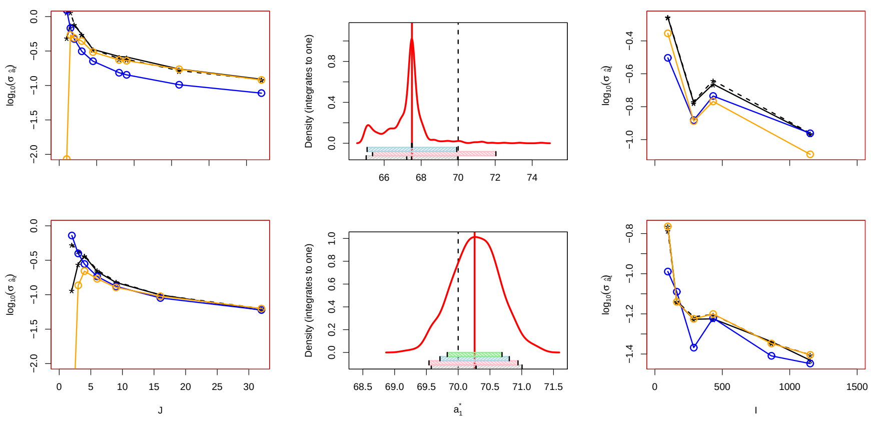


Figure 2: Standard deviation of the bias-parameter estimate $\hat{a}_1$ in model (30) for $C = 2$ and $\log_2$-uniformly distributed replicates as a function of the number of replicates J and as a function of the number of genes I. All estimates are based on the parametric bootstrap method. *Top left*: The standard deviation for $I = 96$ and $J = 2, \ldots, 56$. The solid and the dashed black curves (with asterisks) are the sample standard deviation and MAD of the bootstrap estimates, respectively. The solid orange and blue curves (with circles) are based on the observed and the true Fisher information, respectively. *Top middle*: Kernel density estimate illustrating the distribution of the bootstrap estimates for $I = 96$ and $J = 2$. The solid and dashed vertical lines are the estimated and the true bias parameter, respectively. The horizontal boxes are four different 95% confidence intervals. The bottom two are based on the percentile and bias-corrected bootstrap estimates according to (27). The two on top are based on the true and the estimated Fisher information, where the latter is hardly visible. *Top right*: As in the top left graph, but for $J = 6$ and $I = 96, \ldots, 1152$. *Bottom left*: The standard deviation for $I = 432$ and $J = 2, \ldots, 56$. *Bottom middle*: Distribution of bootstrap estimates for $I = 432$ and $J = 4$. *Bottom right*: The standard deviation for $J = 32$ and $I = 96, \ldots, 1152$.

of genes the former tends to be underestimated. See also Figure 2.

Relative-expression estimates: We have calculated the average coverage of the true relative gene-expression levels $\{\beta_{2,i}\}_i$ for the percentile and bias-corrected bootstrap confidence intervals estimated for each simulation set. The average coverage increases with the number of replicates as depicted in the left graph of Figure 3. The coverage of uniformly and $\log_2$-uniformly distributed probe concentrations are very close to each other.

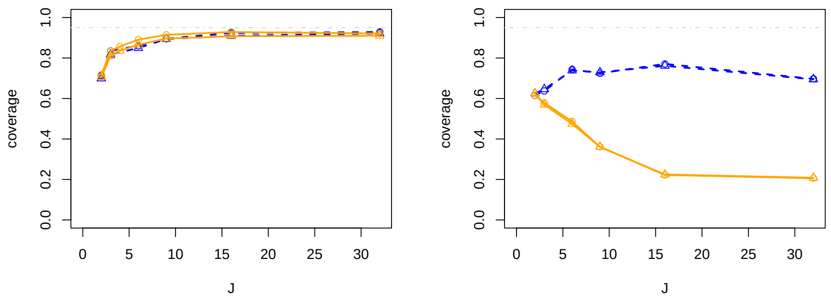


Figure 3: Average coverage (29) of the $\{\hat{\beta}_i\}$ estimates for $C = 2$, $I = 432$ as a function of J. *Left*: Data simulated from model (30). *Right*: Data simulated from model (31). The solid (orange) curves are from $\log_2$-uniform dilution series, and dashed (blue) curves are from uniform dilution series. The curves with triangles are based on the bias corrected (parametric) bootstrap estimates, and the curves with circles are based on percentile estimates.

5.1.2 Misspecified model

Bias-parameter estimates: For the more general model (31), the standard deviation of the parameter estimates based on the estimated Fisher information are still close to those calculated based on bootstrap estimates. See left and right graphs of Figure 4, which depict estimates based on parametric bootstrap samples. However, the standard deviation calculated from the *true* Fisher information is under-estimated with several orders of magnitude, especially for large number of replicates. This discrepancy tends to be large the more replicates there are as in Figure 4. The graphs shown are for $\log_2$-uniformly distributed dilution series. For uniform dilution series the corresponding discrepancy (not shown) increases less rapidly with J, but it still increases (on the log-scale).

Relative-expression estimates: For the misspecified model with $\log_2$*-uniformly distributed dilution series*, the average coverage *decreases* with the number of replicates as illustrated in the right graph of Figure 3. This is because the standard deviation decreases faster than the bias of the relative gene-expression estimates. See also Figure 7. For *uniformly distributed* dilution series, the average coverage is stable or increases slightly.

5.2 Varying number of genes

Bias-parameter estimates: The standard deviation of the bias-parameter estimates for a constant number of replicates decreases with the number of genes as depicted in the right graphs of Figure 2. Observe that the curves in the left panels are smoother than the ones in the right panels. This is because the former ones are estimated with identical sets of expressions levels $\{(\boldsymbol{\beta_i}, \boldsymbol{\gamma_{1,i}})\}_i$ whereas the latter ones are not. Note that since the nonparametric bootstrap estimates resample gene by gene, the corresponding graphs in that case are much smoother (not shown). The results are similar for the misspecified model.

Relative-expression estimates: We find no trend in coverage with respect to the number of genes. Neither for the correct nor the misspecified model.

5.3 Same number of probes

From the graphs in Figure 5, which summarizes the above results, we conclude that the standard deviation of the bias parameters is roughly constant for constant IJ, except when the number of replicates J is small. In that case the large sample approximation is unreliable and the true standard error is larger. We have chosen to display the standard deviation estimated based on the nonparametric bootstrap samples because, as describe above, these are less dependent on the simulated set of gene expressions $\{\beta_{c,i}\}_{c,i}$. Estimates from the parametric bootstrap show a similar but more noisy pattern (not shown). The same behavior is observed (not shown) for the misspecified model.

5.4 Type of replicates

5.4.1 Correct model

Bias-parameter estimates: When it comes to minimizing the uncertainty of the bias-parameter estimates, $\log_2$-uniformly distributed probe concentrations are better than uniformly distributed ones. See left graph in Figure 6, which shows the relative standard deviations, that is, the standard deviation estimates for the $\log_2$-uniform replicates divided by ditto for the uniform replicates. Note that the

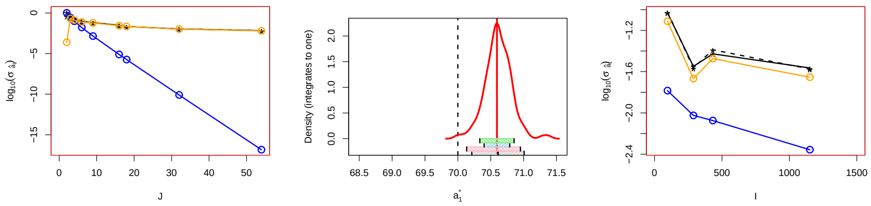


Figure 4: Standard deviation of the bias-parameter estimate $\hat{a}_1$ in model (31) for $C = 2$ and $\log_2$-uniformly distributed replicates (33) as a function of the number of genes I and as a function of the number of replicates J. *Left*: The standard deviation for $I = 96$ and $J = 2, \ldots, 56$. *Middle*: Distribution of the parametric bootstrap estimates for $I = 96$ and $J = 4$. *Right*: As in the top left graph, but for $J = 6$ and $I = 96, \ldots, 1152$.

second factor in (21) goes to $1/4$ when $J \to \infty$ for $\log_2$-uniform replicates (33), and to 1 for uniform replicates (32). Thus, with everything else the same, the relative standard error of these two replicate types is $1/2$ when J is large. By definition both types of replicates are identical for $J = 2$, cf. (32)-(33). No corresponding trend is observed when varying the number of genes.

5.4.2 Misspecified model

A similar comparison for model (31) shows that the relative decrease when the number of replicates grows is even faster. See right graph in Figure 6.

5.5 Estimated gene log-ratios

Define the *estimated* gene log-ratios and the *true* gene log-intensities for channel c and channel 1 as

$$\begin{aligned} \hat{M}_i &= \log_2 \hat{\beta}_{c,i} \\ A_i &= \frac{1}{2} \log_2(z_{c,i} z_{1,i}), \end{aligned} \qquad (34)$$

cf. [14]. Apart from the constant $\log_2(b_c)$, which can be estimated separately, $\hat{M}_i$ is an estimate of the true log-ratio $M_i = \log_2(z_{c,i}/z_{1,i})$ for gene i. In Figure 7, $\{(A_i, \hat{M}_i)\}_i$ are plotted for model (30) with $\log_2$-uniformly distributed replicates

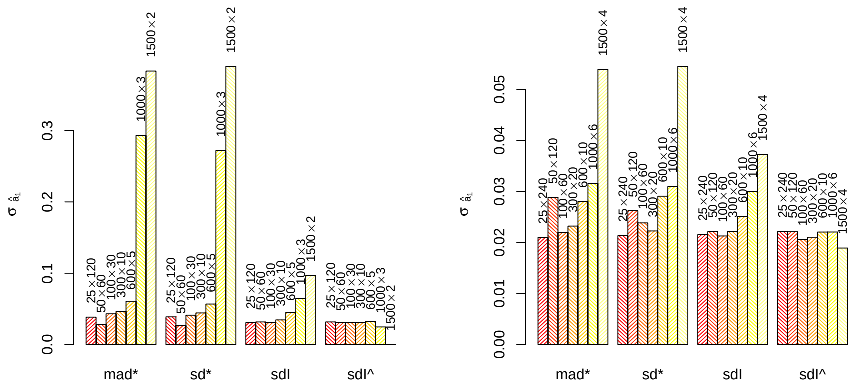


Figure 5: The various standard deviations of the bias-parameter estimates for different array layouts such that IJ is constant. The data was simulated under model (30) with $\log_2$-uniformly distributed replicates (33). Estimates based on nonparametric (26) bootstrap estimates from channel two is shown. *Left*: $IJ = 3000$. *Right*: $IJ = 6000$. Legend: 'sd*' is the standard deviation and 'mad*' is the median absolute deviation of the bootstrap estimates. 'sdI' and 'sdI^' are the standard deviations calculated from the true and the estimated Fisher information, respectively.

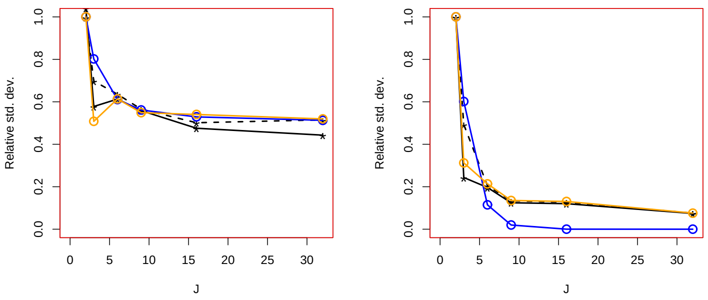


Figure 6: Relative standard deviation between log-uniform and uniform replicates of the estimate of the bias parameter a_1 for $I = 432$ and $J = 2, \ldots, 32$. *Left*: Data simulated from model (30). *Right*: Data simulated from model (31). All estimates are from the nonparametric bootstrap samples.

and $c = 2$, $I = 96$, and for $J = 2$ (right graphs) and $J = 16$ (right graphs). The vertical lines with whiskers depict the 99% bias-corrected confidence intervals obtained from the parameteric bootstrap samples. From these graphs, it is clear that with the two noise structures chosen, the highly differentially expressed genes have a larger standard deviation than the non-differentially expressed genes. Indeed, ignoring the remainder term of (23) and applying Gauss approximation, it follows that

$$V[\hat{M}_i] \approx \frac{V[\hat{\beta}_{2,i}]}{(\ln 2)^2\beta_{2,i}^2} \approx \frac{k_2\sigma_i^2}{(\ln 2)^2 b_2 z_{1,i} z_{2,i}} \left(\frac{\beta_{2,i}}{k_2} + \frac{k_2}{\beta_{2,i}}\right)\left(\sum_{j=1}^{J_i} b_{ij}^2\right)^{-1} \tag{35}$$

with $\ln$ the base e logarithm. The right-hand side is a symmetric function of $M_i = \log_2(\beta_{2,i})$ around $\log_2(k_2)$, where it has its minimum. An estimate of $\beta_{c,i}$ may be negative. This is especially true for highly differentially-expressed genes for which the probe signals in one of the channels are weak, cf. Figure 1. The same is true for the upper and lower limits of the estimated confidence intervals. Since non-positive values are not defined on the logarithm scale, we censor these to equal ∞ or $-\infty$ in the graphs.

5.6 Calibrated probe signals

To illustrate what non-linear effects affine transformations have on the observed relative expression levels $\{y_{c,i,j}/y_{1,i,j}\}_{i,j}$, and how these artifacts are removed by calibration method (19), we calculate the *probe* log-ratios and log-intensities

$$M_{i,j} = \log_2 \frac{y_{c,i,j}}{y_{1,i,j}} \tag{36}$$

$$A_{i,j} = \frac{1}{2}\log_2(y_{c,i,j}y_{1,i,j}) \tag{37}$$

for all probes (i,j) and similarly for the calibrated signals $\{\hat{x}_{c,i,j}\}_{c,i,j}$. As depicted by graphs in Figure 8, the intensity-dependent effects in the raw signals are to a large extent removed by the calibration method. This is true for data both from (30) and (31).

6 Discussion

Calibrated probe signals $\{\hat{x}_{c,i,j}\}_{c,i,j}$ may become non-positive due to uncertainty in the bias-parameter estimates, but also due to the existence of additive noise.

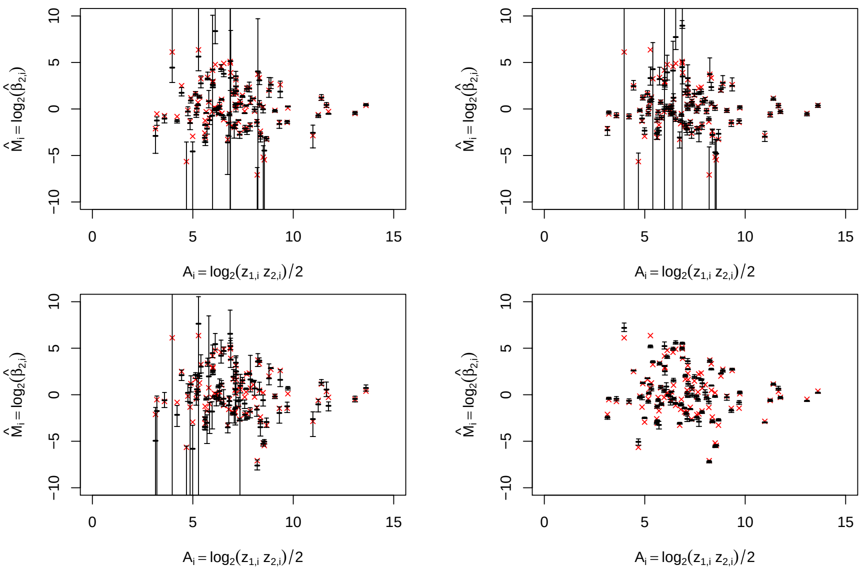


Figure 7: Example of true and estimated *gene* log-ratio versus true log-intensities for $C = 2$, $I = 96$, and varying J. *Left graphs*: $J = 2$. *Right graphs*: $J = 16$. The crosses are the true log-ratios and the thick horizontal lines are the estimated log-ratios with 99% bias-corrected confidence intervals. *Top row*: Data simulated from (30) with $\log_2$-uniformly distributed replicates. *Bottom row*: Same as above, but simulated from (31).

Our model does not try to correct for this. To deal with such problems, see for instance [8]. However, note that the relative expression ratios $\{\beta_{c,i}\}_{c,i}$ are estimated in way which makes the problem of non-positive signals somewhat smaller, although $\hat{M}_i$ still might be undefined for some genes without further restrictions on the $\beta_{c,i}$-parameters.

We have used the calibrated signals $\hat{x}_{c,i,j}$ for estimating probe-specific log-ratios. As an alternative to (34), they may be used for estimating gene-specific log-ratios too, for instance through

$$\hat{M}_i = \log_2 \frac{\sum_{j=1}^{J_i} w_{i,j}\hat{x}_{c,i,j}}{\sum_{j=1}^{J_i} w_{i,j}\hat{x}_{1,i,j}}, \tag{38}$$

where $w_{i,j}$ are weights chosen by the user, for instance $w_{i,j} = 1$ or

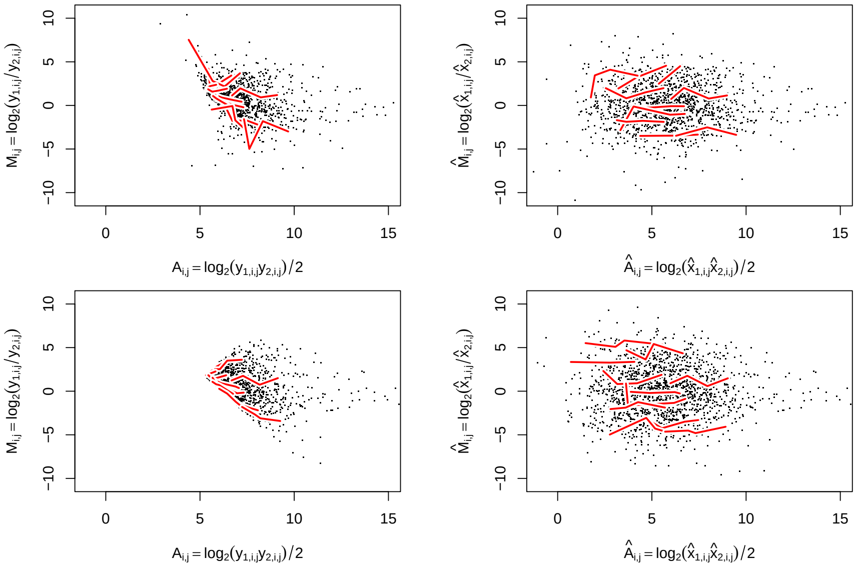


Figure 8: Example of *probe* log-ratio versus log-intensities before (left graphs) and after (right graphs) calibration. Scale parameter b_2 is considered to be known, if not, the left scatter plots are shifted downward. A random set of dilutions series has been highlighted in order to illustrate the fact that the non-linear intensity-dependent artifacts are removed when applying the affine calibration method. *Top row*: Data show simulated from model (30) with $I = 432$ genes, $J = 4$ and $\log_2$-uniform replicates. *Bottom row*: Data show simulated from model (31) for the same array setup.

$w_{i,j} = w(b_{i,j})$, in case the latter are known.

Investigating (35), an interesting alternative error model to simulate from would be the one where, instead of the arithmetical, the geometrical mean of the hybridized dilution concentrations is used, that is, $\bar{z}_i = \sqrt{z_{1,i}z_{2,i}}; \forall i$, cf. (30).

It may even be the case that it is possible that our proposed model can be used for spotted microarrays that contain technical replicates that are not diluted. The reason why this *may* work is that the deposition of spots onto the array surface is imperfect and that different spots may dry differently [4]. This could result in different probe concentrations $\{b_{i,j}\}_j$. Even donut effects may be useful.

7 Conclusions

We have studied a stochastic affine model for scanned microarray data with dilution series and/or spike-ins. Using maximum likelihood techniques estimates of the relative expression levels for each dilution series (gene) follow immediately as well as calibrated signals for all probes. It is important to control the spot concentrations such that no quenching [11] takes place. The model assumes noise that is constant for all probe replicates, but may vary freely between genes. One might generalize the ML estimates to models like (31), although these will be computationally more intensive. However, our investigations show that model (30) is fairly robust against model misspecification. Even though the average coverage (29) of the relative gene-expression levels is too low for the misspecified model, we still see, from the right subplots of Figure 7, that the variability of the estimated log-ratios decreases with increasing J and is satisfactory for large J. It is just the length of confidence intervals for $\beta_{c,i}$ and $\hat{M}_i$ that is too small under model misspecification.

We find that dilution series where the probe concentrations are $\log_2$-uniformly distributed are favorable to the ones that are uniformly distributed, although the average coverage of the log-ratios under the misspecified model is worse for the former; the standard deviation of the estimates is smaller for the $\log_2$-uniform than uniform replicates. Note that the former design is also much easier to implement in practice.

To conclude, we suggest that microarray experiments are designed with multiple dilution series for different genes that are *likely* to be differentially expressed. Although only series with just two replicates are required, we recommend to use more replicates in order to improve the accuracy of the estimates and the calibration as a whole. The method can easily be applied to subgroups of probes, especially spatially disjoint groups such as print-tip groups, which brings up the question on how to optimally position the replicates relative to each other on the microarray. It is beyond this article to give a thorough answer, but it will be a trade-off between accuracy of parameter estimates and size of the regions the estimates should be based on, which in turn depends on the spatial structure of the artifacts.

All analysis was carried out using R [10, 7]. The estimation and calibration methods presented in this paper is made available for free in the aroma package [1].

References

[1] Henrik Bengtsson. aroma - An R Object-oriented Microarray Analysis environment. Preprint in Mathematical Sciences 2004:18, Mathematical Statistics, Centre for Mathematical Sciences, Lund University, Sweden, 2004.

[2] Henrik Bengtsson and Ola Hössjer. Methodological study of affine transformations of gene expression data with proposed normalization method. Preprints in Mathematical Sciences 2003:38, Mathematical Statistics, Centre for Mathematical Sciences, Lund University, Sweden, 2003. Submitted.

[3] Henrik Bengtsson, Göran Jönsson, and Johan Vallon-Christersson. Calibration and assessment of channel-specific biases in microarray data with extended dynamical range. Preprints in Mathematical Sciences 2003:37, Mathematical Statistics, Centre for Mathematical Sciences, Lund University, Sweden, 2003. Submitted.

[4] R.D. Deegan, O. Bakajin, T. F. Dupont, G. Huber, S. R. Nagel, and T. A. Witten. Contact line deposits in an evaporating drop. *Physical Review E (Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics)*, 62(1):756–765, July 2000.

[5] Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.

[6] Wolfgang Huber, Anja von Heydebreck, Holger Sültmann, Annemarie Poustka, and Martin Vingron. Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 1:1–9, 2002.

[7] Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

[8] Charles Kooperberg, Thomas G. Fazzio, Jeffrey J. Delrow, and Thoshio Tsukiyama. Improved background correction for spotted DNA microarrays. *Journal of Computational Biology*, 9:55–66, 2002.

[9] Yudi Pawitan. *In All Likelihood: Statistical Modelling and Inference Using Likelihood*. Oxford University Press, 2001.

[10] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2003.

[11] Latha Ramdas, Kevin R. Coombes, Keith Baggerly, Lynne Abruzzo, W. Edward Highsmith, Tammy Krogmann, Stanley R. Hamilton, and Wei Zhang. Sources of nonlinearity in cDNA microarray expression measurements. *Genome Biology*, 2(11):research0047.1–0047.7, 2001.

[12] David M. Rocke and Blythe Durbin. A model for measurement error for gene expression arrays. *Journal of Computational Biology*, 8(6):557–569, 2001.

[13] David M. Rocke and Stefan Lorenzato. A two-component model for measurement error in analytical chemistry. *Technometrics*, 37(2):176–184, May 1995.

[14] Yee Hwa Yang, Sandrine Dudoit, Percy Luu, and Terence P. Speed. Normalization for cDNA microarray data. In Michael L. Bittner, Yidong Chen, Andreas N. Dorsel, and Edward R. Dougherty, editors, *Proceedings of SPiE*, volume 4266 of *Microarrays: Optical Technologies and Informatics*, pages 141–152, San Jose, California, June 2001. The International Society for Optical Engineering.

A Fisher information

A.1 Fisher information for bias parameters

For simplicity we assume $C = 2$ channels. For any two subvectors $\mathbf{x}$ and $\mathbf{y}$ of $\boldsymbol{\theta}$ and $\boldsymbol{\xi}_i$ we let $\mathcal{I}_{\mathbf{x},\mathbf{y}} = (\mathcal{I}_{x_i,y_j})$ be the $|\mathbf{x}| \times |\mathbf{y}|$ submatrix of $\mathcal{I}$. If $\mathcal{I}_{\mathbf{x},\mathbf{y}} = \mathbf{0}$, $\mathbf{x}$ and $\mathbf{y}$ are orthogonal parameter vectors. It is easy to show that $(\mathbf{a}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ is orthogonal to $(\mathbf{k}, \boldsymbol{\sigma}^2)$. From properties of the inverse Fisher information of structural parameters (see for instance [9], Theorem 9.8) it follows that

$$1/(\mathcal{I}^{-1})_{a_1,a_1} = Q_{1,1} - \frac{Q_{1,2}^2}{Q_{2,2}} \tag{39}$$

$$1/(\mathcal{I}^{-1})_{a_2,a_2} = Q_{2,2} - \frac{Q_{1,2}^2}{Q_{1,1}} \tag{40}$$

where $Q = (Q_{i,j})$ is the 2×2 matrix

$$\begin{aligned} Q &= \mathcal{I}_{\mathbf{a},\mathbf{a}} - \mathcal{I}_{\mathbf{a},\boldsymbol{\beta}\boldsymbol{\gamma}_1} \mathcal{I}^{-1}_{\boldsymbol{\beta}\boldsymbol{\gamma}_1,\boldsymbol{\beta}\boldsymbol{\gamma}_1} \mathcal{I}_{\boldsymbol{\beta}\boldsymbol{\gamma}_1,\mathbf{a}} \\ &= \mathcal{I}_{\mathbf{a},\mathbf{a}} - \sum_{i=1}^{I} \mathcal{I}_{\mathbf{a},\boldsymbol{\beta}_i\boldsymbol{\gamma}_{1,i}} (\mathcal{I}_{\boldsymbol{\beta}_i\boldsymbol{\gamma}_{1,i},\boldsymbol{\beta}_i\boldsymbol{\gamma}_{1,i}})^{-1} \mathcal{I}_{\boldsymbol{\beta}_i\boldsymbol{\gamma}_{1,i},\mathbf{a}}. \end{aligned} \tag{41}$$

where we define $\boldsymbol{\beta}_i = (\beta_2 i, ..., \beta_C i)$, cf. the definition in the "Model" section. One finds that

$$\mathcal{I}_{\mathbf{a},\mathbf{a}} = \sum_{i=1}^{I} \frac{J_i}{\sigma_i^2} \begin{pmatrix} 1 & 0 \\ 0 & k_2^{-2} \end{pmatrix} \tag{42}$$

and the i^{th} term of the right-hand side of (41) equals

$$(\mathcal{I}_{\mathbf{a}\boldsymbol{\beta}_i} \mathcal{I}_{\mathbf{a}\boldsymbol{\gamma}_{1,i}}) \begin{pmatrix} \mathcal{I}_{\boldsymbol{\beta}_i\boldsymbol{\beta}_i} & \mathcal{I}_{\boldsymbol{\beta}_i\boldsymbol{\gamma}_{1,i}} \\ \mathcal{I}'_{\boldsymbol{\beta}_i\boldsymbol{\gamma}_{1,i}} & \mathcal{I}_{\boldsymbol{\gamma}_{1,i}\boldsymbol{\gamma}_{1,i}} \end{pmatrix}^{-1} (\mathcal{I}_{\mathbf{a}\boldsymbol{\beta}_i} \mathcal{I}_{\mathbf{a}\boldsymbol{\gamma}_{1,i}})', \tag{43}$$

where $'$ denotes matrix transposition. Further,

$$\mathcal{I}_{\mathbf{a}\boldsymbol{\beta}_i} = \frac{J_i\bar{\gamma}_{1,i}}{\sigma_i^2 k_2^2}\begin{pmatrix}0\\1\end{pmatrix} \tag{44}$$

$$\mathcal{I}_{\mathbf{a}\boldsymbol{\gamma}_{1,i}} = \frac{1}{\sigma_i^2}\begin{pmatrix}1\\ \beta_{2,i}/k_2^2\end{pmatrix}\mathbf{1}_I \tag{45}$$

$$\mathcal{I}_{\boldsymbol{\beta}_i\boldsymbol{\beta}_i} = \frac{J_i\bar{\gamma^2}_{1,i}}{\sigma_i^2 k_2^2} \tag{46}$$

$$\mathcal{I}_{\boldsymbol{\beta}_i\boldsymbol{\gamma}_{1,i}} = \frac{\beta_{2,i}}{\sigma_i^2 k_2^2}\boldsymbol{\gamma}_{1,i} \tag{47}$$

$$\mathcal{I}_{\boldsymbol{\gamma}_{1,i}\boldsymbol{\gamma}_{1,i}} = \frac{k_2^2+\beta_{2,i}^2}{\sigma_i^2 k_2^2}\mathrm{diag}(1,\ldots,1) \tag{48}$$

with $\bar{\gamma}_{1,i} = \sum_{j=1}^{J_i}\gamma_{1,i,j}/J_i$ and $\bar{\gamma^2}_{1,i} = \sum_{j=1}^{J_i}\gamma_{1,i,j}^2/J_i$. Inserting (44)-(48) into (43), we find, after some calculations based on inversion formulas for block matrices, that the i^{th} term of the sum in (41) equals

$$\frac{J_i}{\sigma_i^2}\begin{pmatrix}1-\frac{\beta_{2,i}^2 S_i}{k_2^2+\beta_{2,i}^2} & \frac{\beta_{2,i}S_i}{k_2^2+\beta_{2,i}^2}\\ \frac{\beta_{2,i}S_i}{k_2^2+\beta_{2,i}^2} & k_2^{-2}-\frac{S_i}{k_2^2+\beta_{2,i}^2}\end{pmatrix}, \tag{49}$$

where

$$S_i = \sum_{j=1}^{J_i}(\gamma_{1,i,j}-\bar{\gamma}_{1,i})^2/\sum_{j=1}^{J_i}\gamma_{1,i,j}^2 = \sum_{j=1}^{J_i}(b_{i,j}-\bar{b}_i)^2/\sum_{j=1}^{J_i}b_{i,j}^2. \tag{50}$$

Inserting (42) and (49) into (41) we obtain

$$Q_{1,1} = \sum_{i=1}^{I}\frac{\beta_{2,i}^2}{k_2^2+\beta_{2,i}^2}\frac{J_i}{\sigma_i^2}S_i = \sum_{i=1}^{I}w_i\beta_{2,i}^2 \tag{51}$$

$$Q_{2,2} = \sum_{i=1}^{I}\frac{1}{k_2^2+\beta_{2,i}^2}\frac{J_i}{\sigma_i^2}S_i = \sum_{i=1}^{I}w_i \tag{52}$$

$$Q_{1,2} = \sum_{i=1}^{I}\frac{\beta_{2,i}}{k_2^2+\beta_{2,i}^2}\frac{J_i}{\sigma_i^2}S_i = \sum_{i=1}^{I}w_i\beta_{2,i}, \tag{53}$$

where w_i are gene-specific weights as in (21). Insertion of (51)-(53) into (40) gives (20) after some manipulations.

A.2 Fisher information for relative expression levels

To begin with we assume known $\mathbf{a}$ and $C = 2$. Then $\beta_{2,i} = \boldsymbol{\beta}_i$ and $\boldsymbol{\gamma}_{1,i}$ are orthogonal to all other parameters and

$$1/(\mathcal{I}^{-1}_{\beta_{2,i},\beta_{2,i}}) = \mathcal{I}_{\boldsymbol{\beta}_i,\boldsymbol{\beta}_i} - \mathcal{I}_{\boldsymbol{\beta}_i,\boldsymbol{\gamma}_{1,i}} \mathcal{I}_{\boldsymbol{\gamma}_{1,i},\boldsymbol{\gamma}_{1,i}} \mathcal{I}'_{\boldsymbol{\beta}_i,\boldsymbol{\gamma}_{1,i}}. \tag{54}$$

Inserting (46)-(48) into (54) it follows after some calculations that

$$1/(\mathcal{I}^{-1}_{\beta_{2,i},\beta_{2,i}}) = \frac{\sum_{j=1}^{J_i} \gamma_{1,i,j}^2}{\sigma_i^2(k_2^2 + \beta_{2,i}^2)} = \frac{z_{1,i}^2 \sum_{j=1}^{J_i} b_{i,j}^2}{\sigma_i^2(k_2^2 + \beta_{2,i}^2)}. \tag{55}$$

A more detailed analysis (not shown here) reveals that additional terms of total size $O(J_\Sigma^{-1})$ are added to the right-hand side of (55) when $\mathbf{a}$ is unknown, and this establishes (23).

Index