Background Correction for
DNA Array Data using a Mixture Model

by

Shan Chang
Master, Wuhan University, 1992
Master, University of Victoria, 2006

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER DEGREE OF SCIENCE

in the Department of Mathematics and Statistics

Background Correction for
DNA Array Data using a Mixture Model

by

Shan Chang
M.Sc., University of Victoria, Victoria, 2006

Supervisory Committee

Dr. Mary Lesperance, (Department of Mathematics and Statistics)
Supervisor

Dr. Min Tsao, (Department of Mathematics and Statistics)
Departmental Member

Dr. Jane Ye, (Department of Mathematics and Statistics)
Departmental Member

Dr. Caren Helbing, (Department of Biochemistry and Microbiology)
Outside Member

Supervisory Committee

Dr. Mary Lesperance, (Department of Mathematics and Statistics)

Supervisor

Dr. Min Tsao, (Department of Mathematics and Statistics)

Departmental Member

Dr. Jane Ye, (Department of Mathematics and Statistics)

Departmental Member

Dr. Caren Helbing, (Department of Biochemistry and Microbiology)

Outside Member

Abstract

There are various kinds of noise that affects DNA array data and it is a challenge to obtain high quality data from an experiment. Background correction is a first step for DNA array data processing; however, current methods for background correction have disadvantages. In this thesis, I propose a mixture model to facilitate the background correction for DNA array data generated by ImaGene. The corresponding software is also provided.

Table of Contents

List of Tables

List of Figures

Acknowledgments

The author would like to thank Department of Mathematics, the University of Victoria for providing a place of study and inspiration. Many thanks are given to Dr. Mary Lesperance for her patience, helpful advice and supervision. I also want to thank Dr. Caren Helbing and her lab researchers for their kindly help and hard work on the experiments where the datasets came from.

Chapter 1, Introduction

1.1 Introduction of DNA array technology

In recent years, DNA array technology is used in many different fields from agricultural to medical sciences. It augments the traditional biological assay methods with small glass chips containing thousands of DNA. DNA array techniques provide the rapid and quantitative analysis of gene expression patterns, patient genotypes, drug mechanisms, and disease onset and progression on a genomic scale. It is a modern technology for measuring the expression levels of a multitude of genes simultaneously (Schena, 2003).

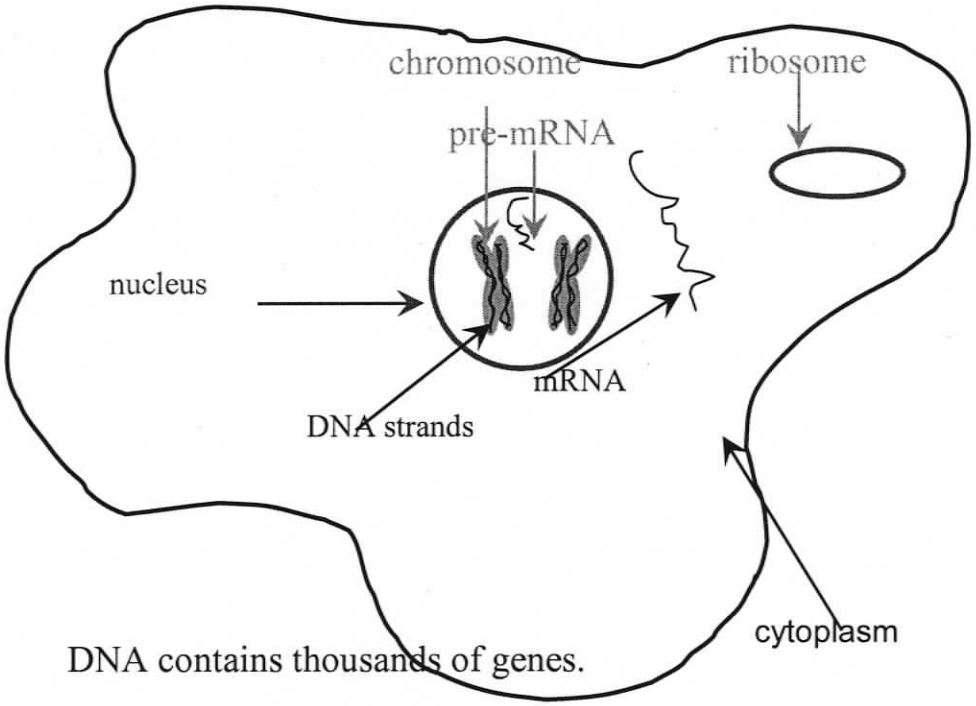

Figure 1-1 DNA and Cell

(http:// www.public.iastate.edu/~dnett/microarray/basicbiology.ppt)

Figure 1-1 shows the simple image of a cell, the DNA strands on a chromosome exist in the nucleus. The protein is synthesized in the cytoplasm. The life information is transferred from DNA by mRNA to create protein. First in the nucleus the pre-mRNA is created with one strand of DNA as the template. Before pre-mRNA enters the cytoplasm it is changed to mRNA. Then mRNA goes to ribosome where the protein is synthesized. In this way, mRNA transfers the genetic information to the protein which plays a great role in cell's life cycle. tRNA and rRNA also play a role in the protein synthesis.

DNA arrays were developed at Standford University in the early 1990s. A DNA array is a small analytical device that allows genomic exploration with high speed, quantity and precision which is unprecedented in the history of biology. The small glass or nylon membrane chips can contain tens of thousands of genes. These chips are used to examine fluorescent or radioactive samples prepared by labelling messenger RNA (mRNA) which is isolated from experimental cells and tissues. When a spot on the chip binds to its complement in the fluorescent/radioactive molecules in solution and is scanned, it will cause the spots to glow. The intensities are proportional to the quantity of mRNA in the sample, or the expression level of each gene. Gene expression is measured by the fluorescence/radioactivity intensity at each spot on the array. Figure 1-2 shows the procedure of a microarray experiment (Schena, 2003).

There are many different types of arrays depending on the techniques and protocol used when generating them, for example: nylon membrane, GeneChip (Affymetrix), Illumina Bead Array, Agilent, Long oblic Ink jet, Comparative Genomic Hybridization (CGH) and DNA macro/microarrays. We concentrate on methods for DNA microarrays,

which contain thousands of genes, and DNA macroarrays which have fewer but larger spots (Geoffrey, 2005).

Unlike traditional methods, the enormous capacity of DNA arrays allows us to analyze the complex gene expression pattern of a sample in a few single experiments. Because patterns of gene expression can have strong correlation with function, DNA arrays can provide unprecedented information on human disease, aging, drug and hormone action, mental illness, diet, and many other clinical matters. DNA arrays can also be used to detect changes in gene expression, starting the new era of genetic screening, testing, and diagnostics. Nowadays, these techniques are adopted in exploring other fields like the genomes of bacteria, viruses, worms, fruit flies, plants, cows, chicken, mice, rats, and primates, etc. The following figure shows the general steps of a DNA array experiment and the consecutive text provides the explanation of these steps (Schena, 2003).

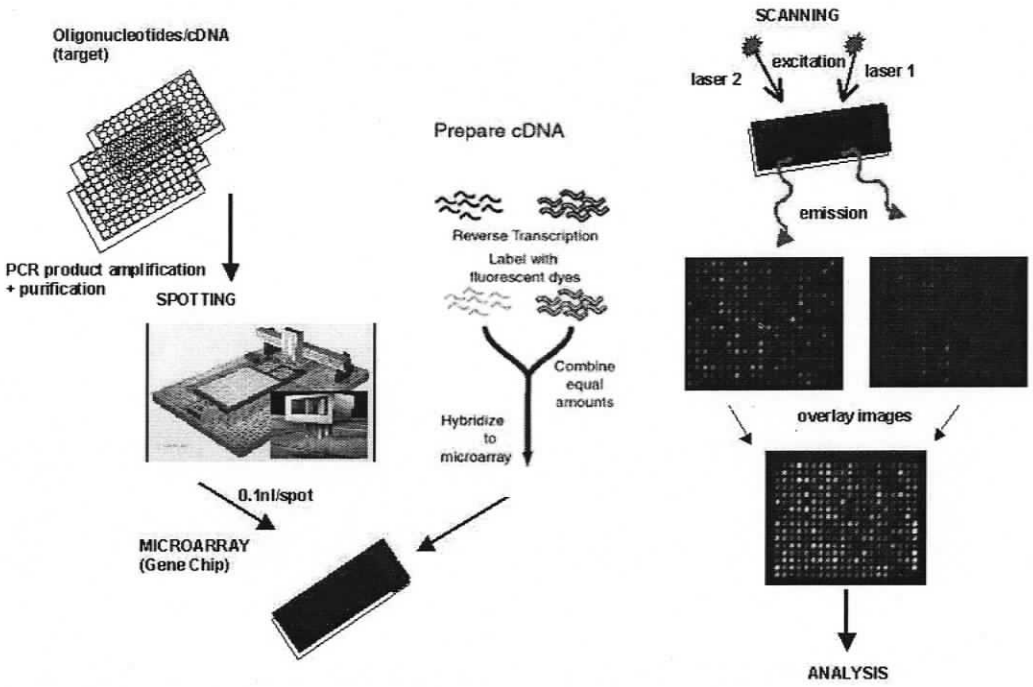

Figure 1-2 A microarray experiment

(http://www.brunel.ac.uk/about/acad/health/healthres/researchareas/mf/microarrayprocedure/)

Different labs have different protocols to create DNA arrays. However, there are common features which are described as follows (Geoffrey, 2005).

- Nucleic Acids: DNA and RNA

DNA is a double helix consisting of two anti-parallel strands, which looks like a twisted long ladder. Both strands of this ladder are weakly connected in the middle by hydrogen bonds. The four bases of those bonds are adenine (A), guanine (G), cytosine (C), and thymine (T). DNA bases pair up with each other, A with T and C with G, to form units called base pairs. If a sequence in one strand is G-T-C-C-T-A, then its complementary sequence in another strand will be C-A-G-G-A-T. When two

complementary sequences find each other--such as the immobilized probe DNA on the array and the mobile target DNA, cDNA, or mRNA in the sample, they will lock together or hybridize.

Complementary DNA, cDNA, is a single-stranded DNA that is complementary to messenger RNA or DNA that has been synthesized from messenger RNA by reverse transcriptase. Complementary DNA is often used in gene cloning, for use as gene probes or in the creation of a cDNA library.

Messenger or mRNA is a copy of the information carried by a gene on the DNA. The role of mRNA is to move the information contained in DNA to the translation machinery. DNA is in the nucleus while proteins are made in the cytoplasm of a cell. RNA transfers the information from the DNA's genetic message to the cytoplasm. To create a DNA array, the RNA is usually isolated from cytoplasm. The purpose of a DNA array is to measure for each gene the amount of message that was transmitted by the RNA. If the RNA finds its complementary part on the array, it naturally binds and sticks to the array. By measuring the intensity on each spot on the array, we obtain information about how much RNA was produced for each gene.

- Array

The array is a solid substrate which is commonly a small piece of glass or nylon, with thousands of spots consisting of cDNA or oligo sequences. Array sizes commonly vary from that of a microscope slide (2.5cm x 7.5cm or 9cm x 12cm) (Schermer,1999) to square silicon chips of 0.5cm x 0.5cm (Warrington et al., 2000). Every spot in the grid of the array can represent an independent experimental assay.

The probe is the DNA of a known sequence that is fixed on the array, while the target is the polynucleotide, eg. cDNA or mRNA, of mixed sequences in the biological sample solution. The target is evaluated indirectly, through its hybridization to the known polynucleotide on the array.

- Spotter

The spotter which is sometimes called the arrayer refers to the robotic instruments for making DNA arrays. The robotic machine is used to fix DNA probes onto a microscope slide or other substrate.

After the DNA is spotted onto the array, it must be fixed onto the surface to prevent the DNA from washing off during the array production. The slide is cleaned and then coated to provide a surface on which the DNA can bind during printing. The other steps in this procedure include air drying and ultraviolet irradiation.

- Labelling a sample for detection

There exist several protocols and commercial kits for the extraction, the purification and the labelling of the target sequences, and their subsequent hybridization with the probe on the DNA array. Normally, an amplification step (PCR) should be considered if the quantity of the experimental material is not adequate.

To identify and measure the presence of a polynucleotide of an unknown sequence in the target sample after it binds to its complement on the DNA array, it is labelled either with a fluorescent dye or radioactive materials. The commonly used DNA array is a 2 color DNA array which uses two dye colors with different wavelengths, normally green (Cyanine 3: Cy3) and red (Cyanine 5: Cy5) and after

scanning creates a 16-bit color digitalized image. Another commonly used DNA array is called "one color macroarray" which uses radioactive labels and after scanning creates a grey scale image.

- Hybridization

Molecule hybridization is the action of single strands of polynucleotides binding to their complement through base-pairing properties to form a complementary double-stranded molecule. This is the chemical process that the labelled nucleic acid molecules use to identify their complementary molecules of known DNA sequence among the spots on the array.

- Scanning/imaging the array

The readout of the DNA array is obtained as a color coded 16-bit TIFF digitalized image using a dual-laser scanner for fluorescence detection for two colour DNA array and 16-bit gray scale image TIFF file for the one color DNA array. The scanner measures the emission intensities at each DNA spot by scanning the DNA array surface. DNA array software such as GenePix (Axon Instruments 2004), ImaGene (BioDiscovery 2004), etc. computes the pixel intensities at each location and provides a data readout as a spreadsheet linked to the image.

The next step in the analysis procedure is to put a grid on the image that allows one to obtain the location of the pixels representing each spot. The image is then processed using quality control procedures, and the software calculates the intensity values of each spot. These values provide a measurement of the amounts of given RNA species in the samples that are represented as the gene expression levels.

The quality operations include grid adjustment, which adjusts the position of grids to match the spots on the array; normalization, which is a method that attempts to remove some of the variation; signal-to-noise adjustments which use the signal-to-noise ratio to estimate the quality of the spot, etc. More details concerning quality procedures are given in Chapter 2.

- Final image processing and verification of DNA array data

  This is a procedure where the biochemist uses other techniques, for example, either visual inspection or other protocols, to remove low quality spots and confirm the findings in a DNA array experiment. The low quality spots are those spots on the array for which the data obtained from the DNA array experiment do not satisfy the quality criteria. This is discussed further in Chapter 2.

Producing a DNA array experiment in a biology laboratory is a long, complex, multi-step process. It involves the mechanical preparation of solutions of DNA clones, PCR amplifications of the DNA, verification of the clone sequences and purification of the probe sequences, spotting of the DNA array slides, hybridization, cleaning, scanning and imaging acquisition, database analysis and verification of the experimental results. These procedures result in a huge dataset and image file. After obtaining the data, the statistical analysis begins. Extracting useful information from a huge dataset is a challenge.

Figure 1-3 shows the DNA array analysis cycle. Briefly, the whole cycle includes probes preparation, DNA array production, sample preparation, scanning, data analysis and biostatistics.

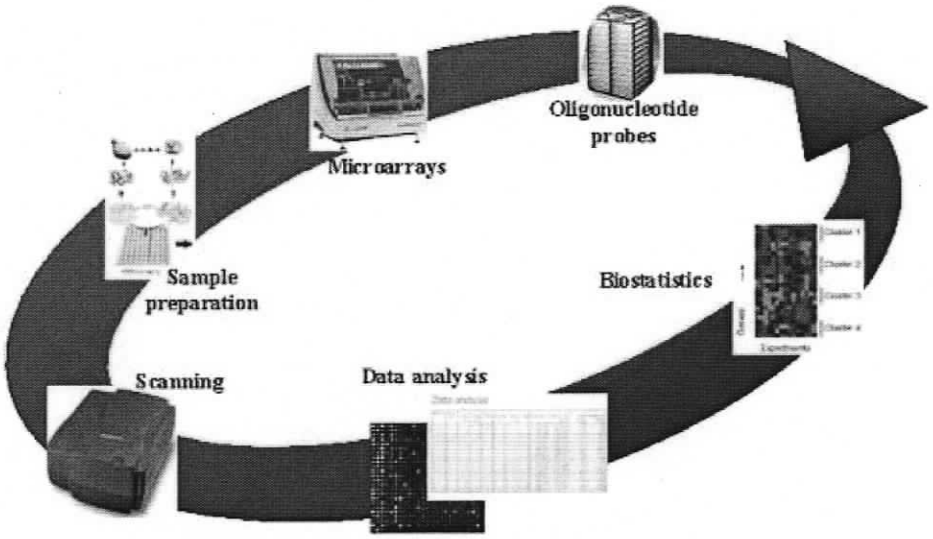

Figure 1-3 DNA array Analysis Cycle

(http://www.microarray.lu/en/MICROARRAY_Overview.shtml)

1.2 Data source

In this thesis, the focus is on the experimental data supplied by Dr. Caren Helbing's lab. In addition, some public datasets are also used to test procedures and to provide comparisons.

1.2.1 Public dataset

At the website (http://genome-www5.stanford.edu/), the Stanford Microarray Database, there are many public datasets contributed by different labs. In this thesis, the yeast dataset (experiment: DES genomic DNA versus MHY26 hpi 1, experiment ID: 16514, slide name: y24n219-2) and the virus dataset (experiment: AD50 hpi 1, experiment ID: 31131, slide name: SHDB 156) are used. The datasets are two color DNA arrays with different meta grids and each meta grid has a different number of rows and columns. The datasets are produced by the GenePix microarray software.

The yeast dataset (16514) has a meta grid (8 x 4) which has 8 row and 4 column sections, and each section has 13 x 20 spots. The total dataset has 8320 spots. The raw dataset is an Excel file.

The virus dataset (31131) has a meta grid (12 x 4) which has 12 row and 4 column sections, and each section has 30 x 30 spots. The total dataset has 43,200 spots. Figure 1-4 shows the image of the virus dataset (31131). The raw dataset is an Excel file.

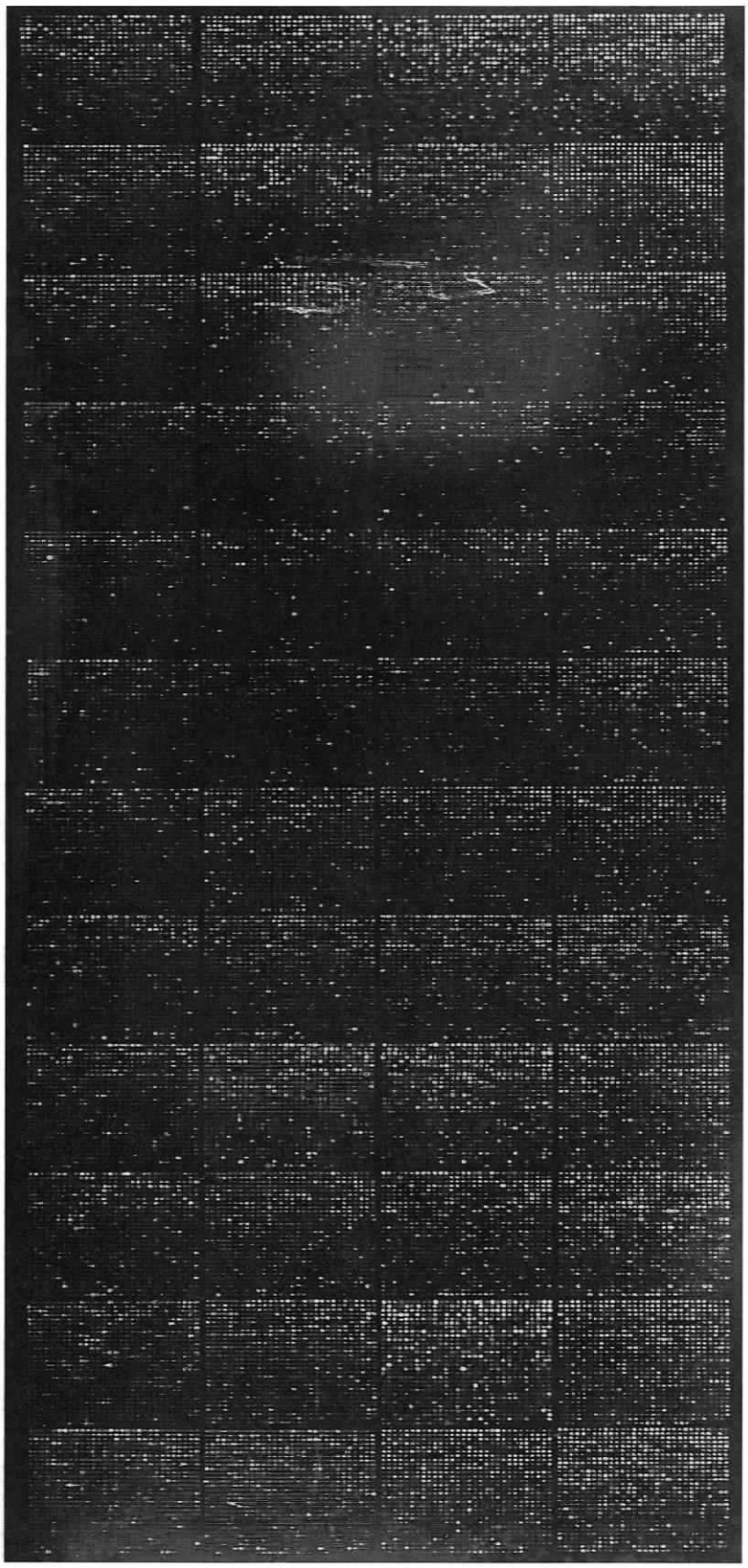

Figure 1-4 Microarray image for the virus dataset 31131

1.2.2 Experimental dataset (From Dr. Caren Helbing's lab)

Dr. Caren Helbing and her lab personnel are studying how thyroid hormones influence cell activity, how chemicals disrupt normal cell activity and how environmental agents could contribute to diseases.

Thyroid hormone is secreted from the thyroid gland which functions as a part of the endocrine system. The endocrine system is a complex ductless gland network that regulates bodily processes. The thyroid hormones are responsible for the development of an organism and the regulation of that organism's metabolic rate. They are very important for human health, especially in babies where they are required for proper brain development. The principal component of this hormone is iodine which is necessary to the body and many problems result if this element is absent (Crump et al. 2002).

Many vertebrates have thyroid glands, including frogs which are studied in Dr. Helbing's lab. Frogs are very sensitive to thyroid hormones which are essential for the metamorphosis of a tadpole into a frog. Metamorphosis occurs as the tadpole tail shrinks and the legs grow. Thyroid hormones cause the tail to destroy itself and for the legs to grow, so frogs can act as indicators for disruption of thyroid hormone action. Evolutionary biologists are interested in how organisms incorporate metamorphosis in their life cycle. Specialists who study cancer are interested in the mechanism by which thyroid hormones signal growth or death. Knowledge of this mechanism may help to kill tumor cells in cancer treatment.

For one experiment conducted in Dr. Helbing's lab, they first prepared cDNA fragments for spotting on the array. The array is prepared as 32 X 42 spots with probes on them. Second they selected test animals and set up the chemical exposure conditions, third used various concentrations of the thyroid hormones such as thiiodothyronine(T3) and thyroxine(T4) in their experiment; fourth isolated RNA and cDNA synthesis from brain, tail and hindlimbs of the frog, fifth made array target sequences and primer designs, sixth prepared total cDNA and amplified and cloned the target using a polymerase chain(PCR) reaction procedure. After hybridization and scanning, the grey scale image was obtained and the related dataset is an Excel format file (Helbing et al. 2006, Zhang et al. 2006).

Figure 1-5 shows a one color macroarray that is used to create a dataset. The dataset is obtained using ImaGene DNA array software. The macroarray here is a meta grid with 32 x 42 spots, so that each array has 1344 spots.

Figure 1-5 Macroarray
(From Dr. C. Helbing's Lab)

Chapter 2, Background Review

2.1 Why background correction?

A common problem in measuring array quantities is the existence of a persistent background signal. It can be observed that even though there are no probes available on some spots, there are still readings at these spots. This phenomenon happens across the array. Scanning software attempts to estimate this background intensity from the regions surrounding the DNA spots and the estimates vary across the array. Background contamination may distort the spot intensity reading, so adjustments should be made to remove this effect (Geoffrey, 2005).

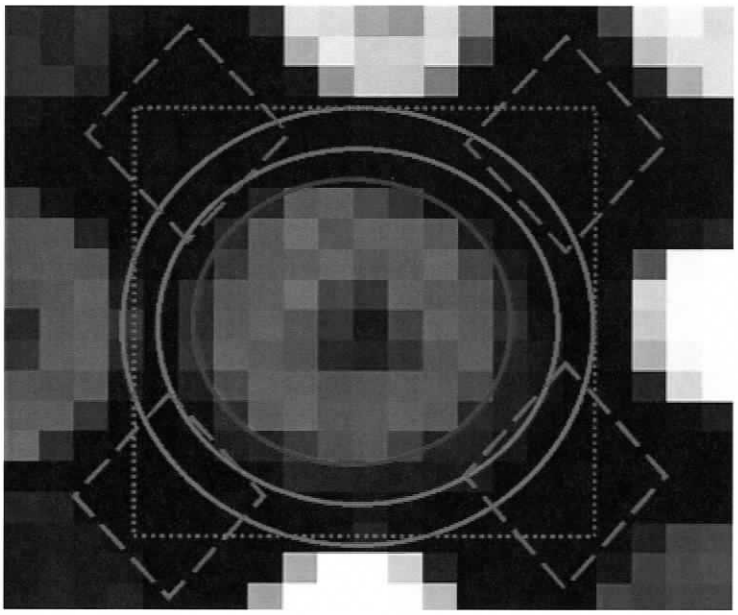

Figure 2-1 Background estimate

(--- *GenePix* --- **QuantArray** --- ScanAnalyze)
(http://www.sinica.edu.tw/~hmwu/Talks)

Different software use different regions of the array to estimate the background intensity. For example, Figure 2.1 shows the spot as the area inscribed by the inner circle which is the same for the three software routines. For the background, the ScanAnalyze software uses the four rectangles near the spot as its background area, while *GenePix* uses the area between the inner circle and the square boundary to estimate the spot's background. QuantArray and ImaGene use the area between the outer two circles to estimate background.

2.2 Existing Methods for background correction

Several solutions have been proposed for background correction. Traditionally, most authors assume an additive model for background, i.e.

$$O = C + T$$

where O is the observed spot foreground intensity, C is the intensity of the background, and T is the target intensity which is a measure of the mRNA in the sample. These three variables are modelled as random variables with means μ_o, μ_c , and μ_t respectively (McLachlan, 2005) .

Unfortunately, the background intensity **within** the spot region cannot be measured directly. The background intensity is estimated from the software using pixel intensities near the spot. Different software provides different estimates of background intensity depending on the region it defines as background (Refer to Figure 2.1) .

The simplest estimate of the target intensity is

$$\hat{T} = O - \hat{C}$$

that is, the estimated local-background intensity $\hat{C}$, is subtracted from the foreground intensity of the spot O, and the result $\hat{T}$ is used for the further analysis (McLachlan, 2005).

There are alternative ways to estimate T. For example, subtract three standard deviations of the background pixel intensities, that is $\hat{T} = O - 3 * \hat{\sigma}_C$; or subtract the average of some spots neighbouring the spot and their local background, $\hat{T} = O - \hat{\mu}_{C+Nbh\_o}$, where NBH_O stands for the neighbouring spots to the current spot and $\hat{\mu}_{C+Nbh\_O}$ stands for the average of the intensities of background of the current spot and background of neighbourhood spots; or subtract the median of some more spots neighbouring to the current spot and their local background which is $\hat{T} = O - m_{C+Nbh\_O}$ where m stands for the median (McLachlan, 2005).

However, the additive assumption, $O = C + T$, is questionable. Note that the above methods for estimating T can lead to a negative estimate of T which is physically impossible. In some DNA arrays, there may be as many as 10% of the genes which might have negative gene expression values using the above methods (Efron et al. 2001).

In practice, some researchers simply discard the negative values from the dataset. Yet, this operation will cause the loss of information and in the some cases, these genes may be of interest, especially in time course experiments where the expression levels vary over time. Other methods are also used, such as subtracting only a fraction (Efron et al. 2001) of the background or some overall small number. However all the above methods make a common assumption that the true value is the sum of two quantities. Each of these quantities can add to the noise, therefore the total variation of the expression level is larger than it should be (McLachlan, 2005).

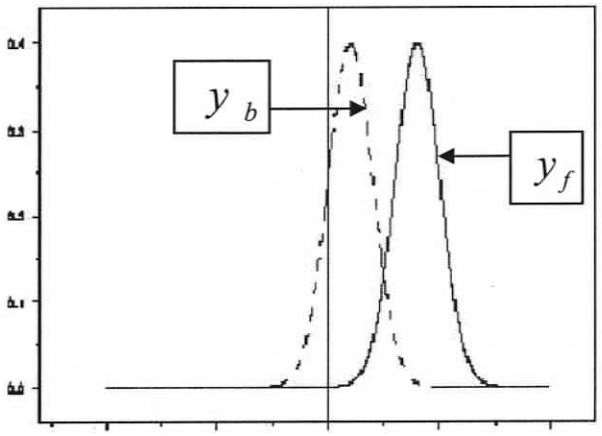

Figure 2-2 Kooperberg et al. model

Kooperberg, et al. (2002) adopt a Bayesian technique to the background correction problem. They assume that for each spot on the array, the average intensity of foreground pixels follows a normal distribution with mean μ_f, and the average intensity of background pixels follows a normal distribution with mean μ_b where $0 \leq \mu_b \leq \mu_f$, conditionally on μ_f and μ_b. They argue that the corresponding sample medians, y_f and y_b are also normally distributed with mean μ_f and μ_b. It follows that

$$T = y_f - y_b$$

has a normal distribution with mean $\mu_t = \mu_f - \mu_b$.

The dashed line in Figure 2-2 is the distribution of y_b which is the median of the background pixel intensities. The solid line is the distribution of y_f which is the median of the foreground pixel intensities.

They use a Bayesian approach, adopting improper non-informative priors on $[0,\infty)$ for μ_t. After obtaining a posterior distribution for μ_t, they use the posterior mean of μ_t given the data as an estimate of the target intensity of a spot. This technique is applied to every spot on the array, including spots whose observed foreground intensity is larger than the observed background intensity.

There are several problems with the Kooperberg et al. (2000) model. The model embodies the assumption that the background and foreground intensities are additive. The second is that the model puts a positive probability on negative values of T, a physical impossibility. The third limitation is the assumption of an improper prior for μ_t. In practice, the mean target intensity will rarely be uniformly distributed.

Parmigiani et al. (2003) suggested skipping background correction altogether, simply regarding foreground intensity as target intensity. This method will yield more conservative ratio estimates especially when the foreground intensities are low. However this method does not take spatial factors into account. Spatial biases might play a great role in the variation of the data and the background and foreground intensity both have such location-dependent effects.

In this thesis, a new method is proposed to solve the problem of background correction. A mixture model approach is used to obtain the nonparametric distribution for the background intensity using the blank spots on the array. A new model is adopted to

describe the interaction of background and foreground, and a method given to estimate the true expression levels. The details of the model are introduced in Chapter 4.

2.3 Data cleaning

Generally speaking, DNA array experiments generate data with errors, which arise from various sources. Such noise may significantly distort the real signal and direct the analysis in the wrong direction. Experimental errors can be classified into two categories: random error and systematic error. Random errors are statistical fluctuations (in either direction) in the measured data due to the precision limitations of the measurement device. Random errors usually result from the experimenter's inability to take the same measurement in exactly the same way to get exact the same number. Systematic errors, by contrast, are reproducible inaccuracies that are consistently in the same direction. Systematic errors are often due to a problem which persists throughout the entire experiment. Some errors can be avoided by carefully designing the experiment but some errors are unavoidable. Obtaining data with satisfactory precision and accuracy has been one of the biggest challenges in the application of DNA array technology.

Before any further statistical analysis, data cleaning is necessary to reduce the noise as much as possible to lessen their influence on the analysis.

2.3.1 The reasons for noise

Generally each procedure of the DNA array experiment will cause errors. Some can be removed by carefully designing the experiment and accurately taking the measurement

or using powerful equipment. Others can be reduced by statistical methods or some ad hoc methods. Table 2.1 below extracted from McLachlan et al 2005 gives the sources of variation in a DNA microarray experiment.

Table 2-1 Sources of variation in a DNA arrays experiment

No	DNA array Procedure	Source of Variance
1	mRNA	Differences in conditions
		Differences between experimental subjects within the same covariate level
		Differences between samples from the same subject
		Variation in mRNA extraction methods from original sample
		Variations in reverse transcription
		Difference in PCR amplification
		Different labelling efficiencies
2	DNA array production	Print-pin anomalies
		Variations in printed probe quantities even with the same pin
		Chip batch variation (due to many sources of unknown variations)
		Differences in sequence length of the immobilized DNA
		Variations in chemical probe attachment levels to the slide
3	Hybridization process	Different dye sensitivities
		Inequalities in the application of mRNA to the slide
		Variations in the washing efficiencies of non-hybridized mRNA off the slide
		Other differences in hybridization parameters, such as: - temperature - experimenter - time of the day
4	Scanning	Different scanners
		Different photo-multipliers or gain
		Different spot-finding software
		Different grid alignments

2.3.2 Introduction to quality factors

Even though an experiment is carefully designed, variation still exists in the results. ImaGene (BioDiscovery, 2004) produces some so-called quality factors which measure

the quality of the data. Below is a description of the quality factors which will be used in this thesis. At the end of this section, the ImaGene variables which will be used in the model are also listed.

Within ImaGene, the user can use various types of automated spot flagging schemes to help locate poor quality spots. There are tools to set flags for low-expressed spots, missing spots or negative spots. ImaGene has a complex tool to define low-quality spots. This tool includes seven different criteria for spot fail/pass tests that can be used in combination as well as separately. There are a set of seven flags to let the software determine whether these seven criteria are satisfied or not. If one criterion is not satisfied, the respective flag is set to one. The details of this quality control mechanism are given below. ImaGene has a menu for the user to set the flag values.

- Flag for background contamination (BC)

  ImaGene uses a t-test to measure whether the background is contaminated. The background level of each individual spot is compared to the local background distribution over the image. The resulting p-value is subtracted from 1. If 1 minus the p-value exceeds the given threshold, then the flag is set to 1 which means the background is contaminated. The threshold can be set to any value between 0.5 and 1, and the default value is 0.9995. For example, for each spot S, let $y_{b,s}$ be the median of the background intensity pixels at spot S, $\bar{y}_b$ and s_b are the mean and standard deviation of the $y_{b,s}$ over the array, then

$$t_{b,s} = \frac{y_{b,s} - \bar{y}_b}{s_b / \sqrt{n}}$$

  where n is the total number of pixels in spot S. The BC value is

$$BC = 1 - P\{|t_{n-1}| > t_{b,s}\}$$

- Flag for foreground contamination (FC)

  This flag is similar to the background contamination flag. For every spot, the foreground variance is compared to the distribution of signal variance measurements across the image and a t-test is computed. If 1 minus the p-value exceeds the threshold, then the flag is set to 1 which means the foreground is contaminated. The threshold can be set to any value between 0.5 and 1, and the default value is 0.9995. For example, for each spot S, let $y_{f,s}$ be the median of the foreground intensity pixels at spot S, $\bar{y}_f$ and s_f are the mean and standard deviation of the $y_{f,s}$ over the array, then,

$$t_{f,s} = \frac{y_{f,s} - \bar{y}_f}{s_f / \sqrt{n}}$$

  where n is the total number of pixels in that spot. The FC value is

$$FC = 1 - P\{|t_{n-1}| > t_{f,s}\}$$

- Flag for high ignored percentage (HIP)

  When ImaGene processes the scanned image, some pixels may be assigned neither to foreground nor to background if local contamination exists. Such contamination is excluded from the measurement and treated as an "ignored" region. If the ratio of ignored pixels is higher than the designated threshold which has the default value of 25%, then the flag is set to 1 which indicates poor quality of the spot. The ratio is defined as follows

$$HIP = \frac{I}{I+S} \times 100\%,$$

where S stands for the signal area and I stands for the ignored area, each measured in pixels.

- Flag for high open perimeter percentage (OPP)

  This flag is set to 1 when the spot's deformation exceeds the pre-set threshold, where the default value is 25%. The percentage of signal perimeter pixels that touch the border of the rectangle that is expected to contain the spot is computed, i.e. the spot will have a large OPP if it is far away from its expected position or it has an abnormal form (Refer to Figure 2-3).

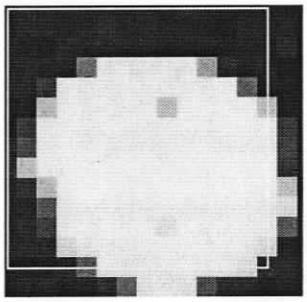

Figure 2-3 Example to show the open perimeter percentage

The open perimeter percentage is defined as

$$OPP = \frac{O}{S} \times 100\%$$

where O stands for the number of pixels which are on or outside of the expected rectangle line, S stands for the number of pixels on the spot perimeter.

- Flag for abnormal shape regularity (ASR)

  Ideally the spot is supposed to be a circle, but generally there are deformations and it is not a perfect circle. This flag is set to 1 as low quality when the shape regularity is below the pre-set threshold where the default value is 0.65. The ASR value is computed as one minus the ratio of number of non-signal pixels that fall

within the inscribed circle of the spot over the circle's area in pixels (Refer to Figure 2.4).

Figure 2-4 Example to show the abnormal shape regularity

The shape regularity index is defined as

$$ASR = 1 - \frac{NS}{S}$$

where NS stands for the number of non-signal pixels in the above circle, and S stands for the signal area in pixels.

- Flag for low Area-to-Perimeter ratio (APR)

  This measure is quite similar to the shape regularity index ASR. This ratio is the area of spot over the square of the spot perimeter, and multiplied by 4π . The flag is set to 1 when the ratio is less than the pre-set threshold where the default value is 0.65. That is, the ratio of Area-to-Perimeter is defined as:

$$APR = \frac{S}{P^2} \times 4\pi$$

where S stands for the spot area, P stands for the perimeter of the spot, both in pixels. If the shape of the spot is a perfect circle, then

$$APR = \frac{\pi r^2}{(2\pi r)^2} \times 4\pi = 1$$

2.3.3 List of useful variables from ImaGene

Table 2.3 shows the name of data fields from ImaGene and their respective variable names in this thesis. The flag, second variable in Table 2.3, takes on values from 0 to 7 according to the combination of the above quality factors. Table 2.2 lists the flag values together with the condition corresponding to the flag values. By default, the user can tell the software to use specific quality measures when setting the flag. Spots that have BC greater than 0.9995, FC>0.9995, HIP >35%, OPP>25%, ASR<0.65, and APR<0.65 are considered to be of low quality. These are the default settings. For the analysis in this thesis, the spot is considered bad and removed from the dataset if the flag value is 3 or 5. All areas are measured in units of number of pixels.

Table 2-2 Flag values in ImaGene

Flag	Condition
0	No flag
1	No particular reason
2	Empty spots
3	Low quality spot(at least one of the criteria fails)
4	Negative spots
5	Empty spots
6	Poor spots
7	Negative Spot

ImaGene has flags for two colour arrays. The values 2,3,4 are automatically generated for the green color, and the values 1, 5, 6 and 7 are manually created using the flag tools and they are for the red color. Dr. Helbing's datasets have only one "color" and they use flags 3 and 5 to indicate low quality spots.

Table 2-3 Table of variables used in the model and software

No	Measurement Variable in ImaGene	Variable in thesis
1	Gene name or gene ID	labels
2	Flag(quality value)	flag
3	Signal mean	xf
4	Background mean	xb
5	Signal median	yf
6	Background median	yb
7	Signal Area	nf
8	Background Area	nb
9	Signal stand deviation	sf
10	Background stand deviation	sb
11	Shape Regularity	ASR
12	Ignored Area	HIP
13	Spot Area	spotArea
14	Area to Perimeter ratio	APR
15	Open Perimeter ratio	OPP
16	Background contaminated flag	BC
17	Signal contaminated flag	FC

2.3.4 Saturated Spots

Generally the intensity of a spot is proportional to the amount of mRNA binding to the spot. When the intensity of a spot reaches a maximum and does not increase any further, we say that the intensity of the array spot is saturated. Saturated spots may confound the interpretation of DNA array data. They can mask differences in the expression of the corresponding genes between experimental conditions. Also debris on the array or other imperfections of the array might also cause spots with intensities over 50,000 (Schena, 2003).

In practice, scientists often remove the saturated spots from the dataset because they may not provide useful information about the gene. Moreover, the saturated spot might bleed and affect the reading of neighborhood spots and the background nearby. In

the proposed analysis, the saturated spots and their neighborhood spots are removed in the data cleaning step. Some researchers consider the spots as saturated spots if their intensities values are over 50,000.

Chapter 3, Study on datasets

Each dataset generated by GenePix or ImaGene has variables for Gene Name, Flag, Background/Foreground mean, Background/Foreground median, the location of the spot, Background/Foreground area, and Background/Foreground standard deviation. GenePix gives the coordinates of the top right and bottom left of the box which contains the spot. ImaGene gives the coordinates of the center of the spot. Both of these software use pixel as the measurement unit. ImaGene provides more quality factors which were introduced in Chapter 2. We define the origin point as the top left corner, and the row coordinates going down, and the column coordinates increasing to the right. The box here is defined as an expected rectangle surrounding the spot. Variable names can be changed by the software user.

Table 3.1 is a list of variables produced by GenePix that are used in the subsequent analysis. Channel 1 is red and channel 2 is green.

Table 3-1 Variables in the Genepix datasets

(no channel two and unused variables)

No	Data Column	Definition
1	SPOT	The serial number of spot on the array
2	GENE NAME	Gene Name
3	SECTORROW	Row number in subgrid(Sector)
4	SECTORCOL	Column number in subgrid(Sector)
5	SECTOR	Serial number of meta grid
6	CH1I_MEAN	Signal Mean for channel 1
7	CH1I_MEDIAN	Signal Median for channel 1
8	CH1B_MEAN	Background Mean for channel 1
9	CH1B_MEDIAN	Background Median for channel 1
10	CH1I_SD	Signal stand deviation of channel 1
11	CH1B_SD	Background stand deviation of channel 1
12	TOT_SPIX	Total number of pixels in signal area
13	TOT_BPIX	Total number of pixels in background area
14	Top (in pixels)	The row coordinate of the top-left of the box
15	Left (in pixels)	The column coordinate of the top-left of the box
16	Bottom (in pixels)	The row coordinate of the bottom-right of the box
17	Right (in pixels)	The column coordinate of the bottom-right of the box
18	FLAG	Flag set for quality control
19	Diameter	The diameter of the spot

Table 3.2 is a list of variables produced by ImaGene that are used in the subsequent analysis.

Table 3-2 Variables in the ImaGene datasets omitting unused variables

No	Data Column	Definition
1	Meta Row	Serial no for Meta Grid Row (=1 if only 1 Meta Grid)
2	Meta Column	Serial no for Meta Grid Column (=1 if only 1 Meta Grid)
3	Row	Row number in Meta Grid
4	Column	Column number in Meta Grid
5	Gene ID	Gene Name
6	FLAG	Flag set for quality control
7	Signal Mean	Signal Mean
8	Background Mean	Background Mean
9	Signal Median	Signal Median
10	Background Median	Background Median
11	Signal Mode	The mode of the signal
12	Background Mode	The mode of Background
13	Signal Area	Total number of pixel of signal area
14	Background Area	Total number of pixel of background area
15	Signal Std	Standard deviation of signal
16	Background Std	Standard deviation of background
17	Shape Regularity	Quality factor evaluate the shape regularity of signal
18	Ignored Area	Quality factor measure the ignored area (in pixel)
19	Spot Area	Total number of spot area (=signal area)
20	Area To Perimeter	Quality factor measure the regularity of the spot
21	Open Perimeter	Quality factor measure the regularity of the spot
22	XCoord	Coordinate of row in pixel for the center of the spot
23	YCoord	Coordinate of column in pixel for the center of the spot
24	Diameter	The diameter of the spot (in pixel)
25	Offset Position	Measure the offset of the spot away from the expecte place

3.1 Distribution of pixel intensity

First we check the distribution of the pixel intensities. Here Matlab is used to extract the pixel information from the array image file using the location of the spots. Different types of spots, such as saturated spots, low expressed spots, and blank spots are chosen to determine the distributional properties. The procedure is given below.

The public data used here is experiment ID 31131 downloaded from SMD, Stanford's Microarray Database (http://genome-www5.stanford.edu/). The dataset is

created by GenePix and is stored as an Excel file. There is a corresponding image file which is stored in GIF format. The data has 12 X 4 sub-grids and each sub-grid has 30 X 30 spots. One can select the spot by designating the section, row and column either from the image data file or from the Excel data file.

The dataset includes the spot location on the array (in pixels), so that one can use image software to obtain the pixel intensity. The original image is made up of two channels of data. CorelDraw 10.0 can change the color image to 16-bit greyscale. Here Matlab and S-Plus are used to obtain the distribution of the spot pixel intensities. The spot pixel information is obtained using Matlab6.5 with functions imread(), imshow(), imwrite(), ind2gray(), and im2uint16(). The last two functions are not needed if the image file is a greyscale image file. Refer to Appendix 1 for function descriptions.

First Matlab is used to extract only one channel (Red) from the original image and then saved it to TIF format. Then CorelDraw is used to change the TIF file to 16-bit greyscale. The Matlab function ind2gray() may also be used to do this transformation directly (Refer to Appendix 1 for details.).

After the picture is changed to a 16-bit greyscale image file (TIF format), under Matlab, one can use the imread() function to obtain the image object. Using the raw data (Excel) file and the GIF image file, one can select a desired spot and then obtain the pixel intensities of the spot from the image object using imread(). For example, the Gene ITGB2 has location in section 1, row 1 and column 1. Its spot box is the rectangle which has coordinates in pixels, as LeftTop(34,31) and RightBottom(47,44), where the first coordinate refers to column, and the second coordinate refers to the row. We read its pixel intensities from the image object defined by the rectangle (31:44, 34:47). We

exported the data to S-Plus to obtain histograms and a QQ plot of the pixel intensities for the spot.

For Dr. Helbing's data, the following steps were adopted. The image file of the array and dataset from her lab were produced and saved using ImaGene. We used one sample file as an example, T3 0.31ppb 24 hour, sample 14, abbreviated as t3h24s14. The variable flag incorporated the quality measures that were pre-selected by the user. Here, the user selected only two quality factors: background contamination (BC) and foreground contamination (FC) (See Chapter 2, section 2.3.2 for the definitions). The other quality factors were not used to set the variable called flag. We can still use the other quality measures in the later analysis.

Various types of spots including blank spots, spots with low quality and saturated spots were chosen to investigate the pixel intensity distribution. In this thesis, only four of them are included although many more were studied. There are discrepancies with the diameter and coordinate values according to the ImaGene software manual, and some processing is needed to be consistent with the image file. The steps for the abstraction of pixel information are as follows:

1) Find the spot of interest in the data file, such as a blank spot, a saturated spot, etc. Record the relative row and column (Row, column) from Table 3.1 which identifies its location on the array.
2) Find the XCoord, YCoord which gives the center of each spot in pixels. Adjust the coordinates by dividing by 3. This was determined by analyzing the ImaGene dataset and is consistent with the actual image size. There are 42 spots in a row and 32 spots in a column. Since the entire image size is 660 X 500 pixels, the spot

size cannot be greater than 15 X 15 pixels. However, the diameter reading provided by image for each spot is approximately 27.5 pixels, implying the whole image size should be greater than 800 X 1100 pixels. We considered adjusting the pixel sizes by dividing by 2 and 3; dividing by 3 produced the most plausible results. The pixel measurements were therefore all scaled by 3 to obtain plausible values.

3) Start Matlab6.5 and use imread() to read the TIF or GIF file to the workspace. This step will create two matrices. Loc is the location of each pixel, the other, Map saves the color information. If the image file is greyscale, this matrix is null. Then use the ind2gray() and im2uint16() to change the file to a greyscale file. Refer to Appendix 1 for details.

4) Obtain the relative box image using

   R= Diameter/2

   sp1=Loc(XCoord-R:XCoord+R, YCoord-R:YCoord+R)

5) The function imshow(sp1) displays this rectangular in Matlab. Since it is too small, use imresize(sp1,10) to enlarge it 10 times before displaying. Then use File->Export to save the image file to jpg or another image file format, such as TIFF, GIF, etc.

6) Obtain the spot matrix sp1 from Matlab and copy it to S-Plus and change the matrix to a vector after removing the corner pixels. The S-Plus functions are used to obtain histograms and QQ plots.

Data quality statistics like Area-to-Perimeter(APR), Shape Regularity(ASR), Open Perimeter(OPP) and High Ignored percentage(HIP) are calculated by the S-Plus program,

backsub.ssc given in Appendix: 3, when reading the data to S-Plus and checked at this time.

The following table gives a list of the spots and their plot index for spots from SMD and the Helbing lab. The plots were produced using the above procedure. In Table 3-3 we use a triple coordinate (Sector, Row, Column) to refer a spot.

Table 3-3 Index for the plot of pixel intensity

No	Source	Gene Name	Spot	ASR	OPP	APR	HIP
1	Pub:31131	ITGB2	(1,1,1)	-	-	-	-
2	Pub:31131	Image 1882829	(8,26,30)	-	-	-	-
3	Pub:31131	Image301627	(8,29,18)	-	-	-	-
4	Pub:31131	Blank	(33,27,1)	-	-	-	-
5	Helbing Lab: t3h24s14	Blank	(1,27,27)	0.9611	0	1	0
6	Helbing Lab: t3h24s14	C-mos	(1,1,1)	0.9805	0	1	0.65%
7	Helbing Lab: t3h24s14	ING2 exon 2	(1,1,38)	0.961	0	1	28.73%
8	Helbing Lab: t3h24s14	NEDD 4	(1,21,28)	0	0.3247	0.3529	0.13%
9	Helbing Lab: t3h24s14	Larval b I globin	(1,8,29)	0.8514	0.19	1	0

Using Matlab functions described in the Appendix 3 to extract the intensities of the spot from the dataset, the aim is to study the distribution of the intensity and the influence of the quality factors and how to combine the quality measures in this thesis.

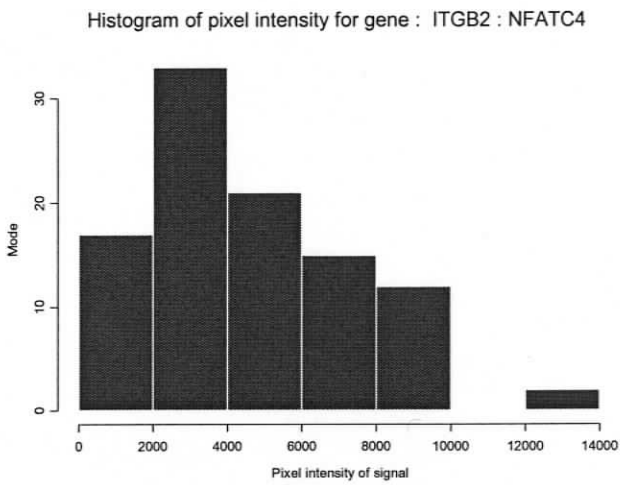

Figure 3-1 Histogram of pixel intensity for Spot 1

Figure 3-1 shows the histogram of pixel intensities for Spot 1. It is created using the greyscale intensities transferred from the red colour image.

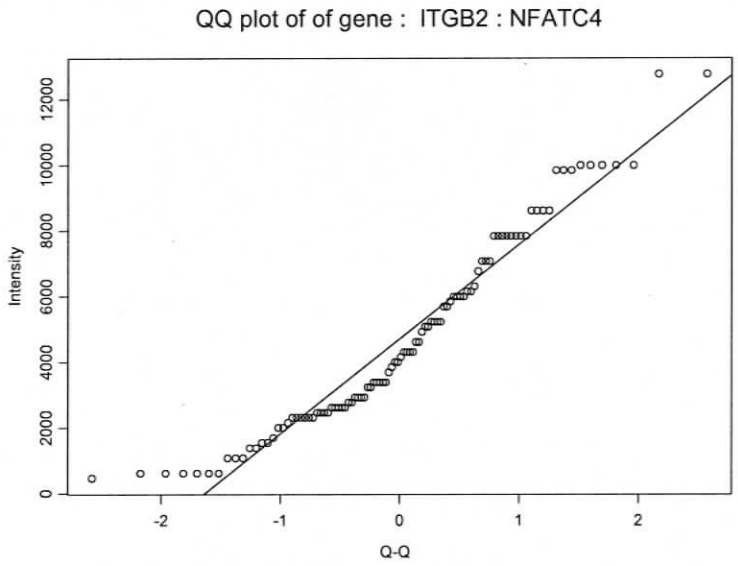

Figure 3-2 Normal QQ plot of pixel intensity for Spot 1

Figure 3-2 shows the QQ plot of pixel intensities of Spot 1 when its image was changed to greyscale. It shows roughly a normal distribution with some outliers.

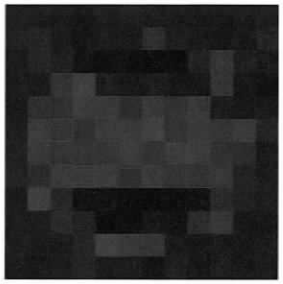

Figure 3-3 Spot image in red channel only for Spot 1

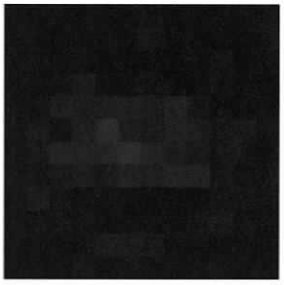

Figure 3-4 Spot image in grey scale for spot 1

Figure 3-3 shows the Spot image in red channel while Figure 3-4 shows when the image is changed to a greyscale image.

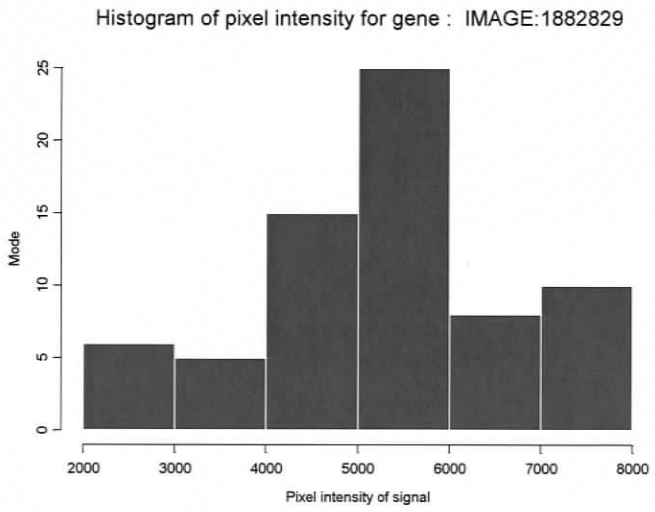

Figure 3-5 Histogram of pixel intensity for Spot 2

Figure 3-5 shows the histogram of pixel intensities for Spot 2 when its image was changed to greyscale.

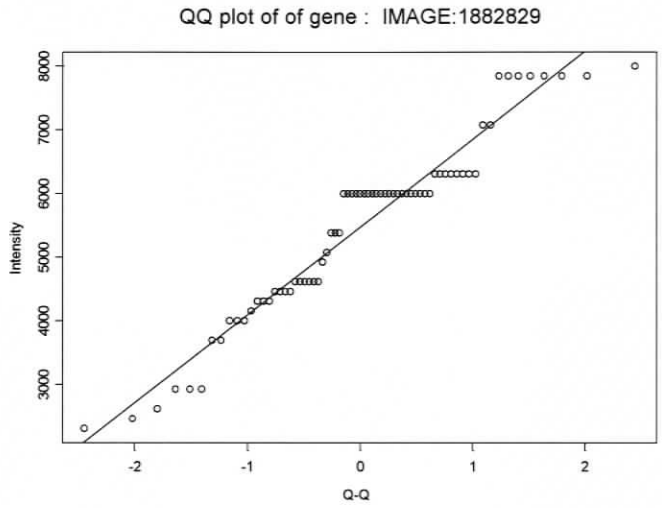

Figure 3-6 Normal QQ plot of pixel intensity for Spot 2

Figure 3-6 shows the QQ plot of pixel intensity of Spot 2 when its image was changed to greyscale. It shows that the distribution is approximately normal.

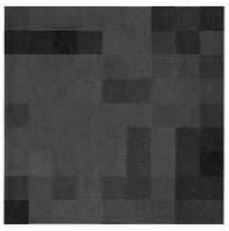

Figure 3-7 Spot image in red channel only for Spot 2

Figure 3-8 Spot image in grey scale for Spot 2

Figure 3-7 shows the Spot image in the red channel while Figure 3-8 shows when the image is changed to greyscale for Spot 2.

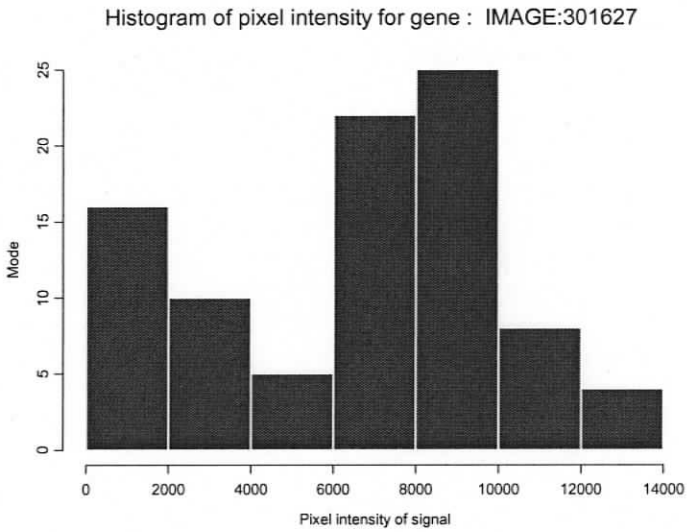

Figure 3-9 Histogram of pixel intensity for Spot 3

Figure 3-9 shows the histogram of pixel intensities for Spot 3, it is created by the greyscale intensities transferred from the red colour image.

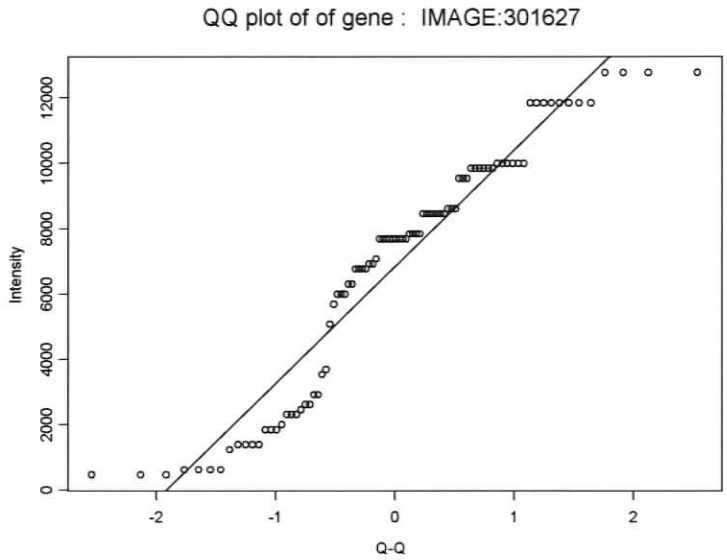

Figure 3-10 Normal QQ plot of pixel intensity for Spot 3

Figure 3-10 shows the QQ plot of pixel intensity of Spot 3 when its image was changed to greyscale. It is not likely normally distributed.

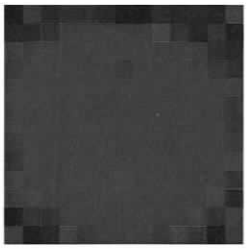

Figure 3-11 Spot image in red channel only for Spot 3

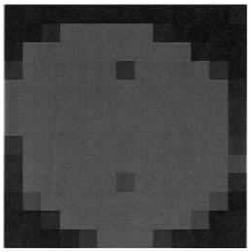

Figure 3-12 Spot image in grey scale for Spot 3

Figure 3-11 shows the Spot image in red channel while Figure 3-12 shows when the image is changed to greyscale for Spot 3.

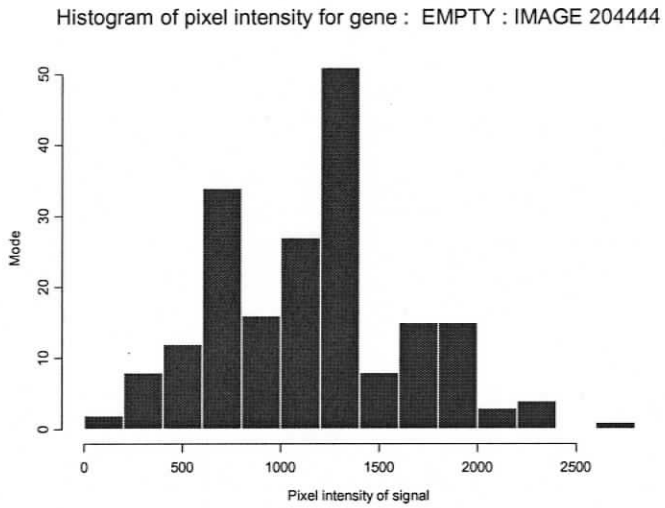

Figure 3-13 Histogram of pixel intensity for Spot 4

Figure 3-13 shows the histogram of pixel intensities for Spot 4, it is created by the greyscale intensities transferred from the red colour image.

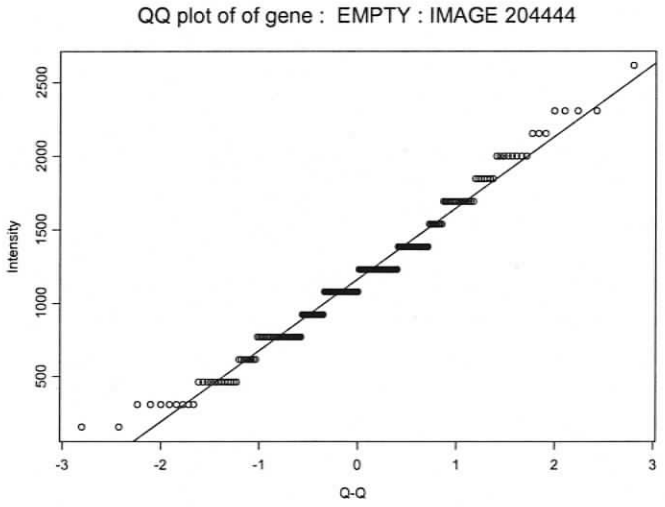

Figure 3-14 Normal QQ plot of pixel intensity for Spot 4

Figure 3-14 shows the QQ plot of pixel intensity of Spot 4 when its image was changed to greyscale. It looks like a normal distribution.

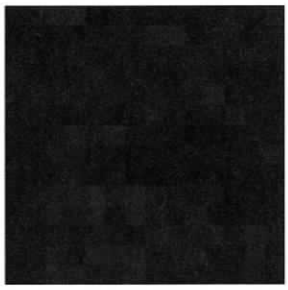

Figure 3-15 Spot image in red channel only for Spot 4

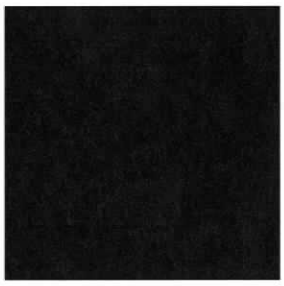

Figure 3-16 Spot image in grey scale for Spot 4

Figure 3-15 shows the Spot image in red channel while Figure 3-16 shows when the image is changed to greyscale image for Spot 4.

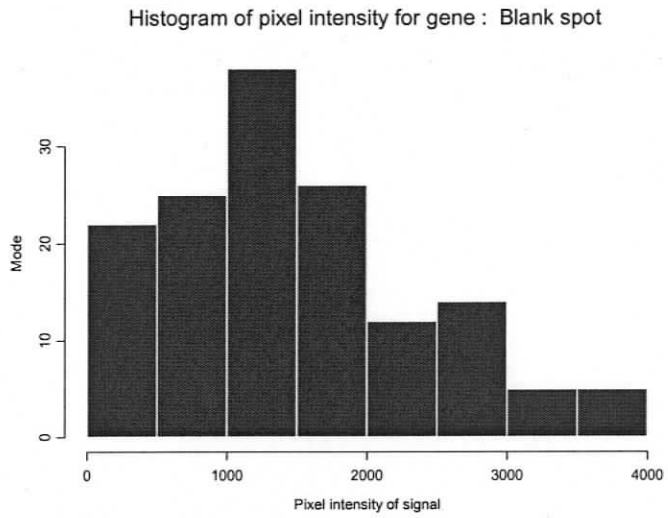

Figure 3-17 Histogram of pixel intensity for Blank Spot 5

Figure 3-17 shows the histogram of pixel intensities for Spot 5 which is a blank spot.

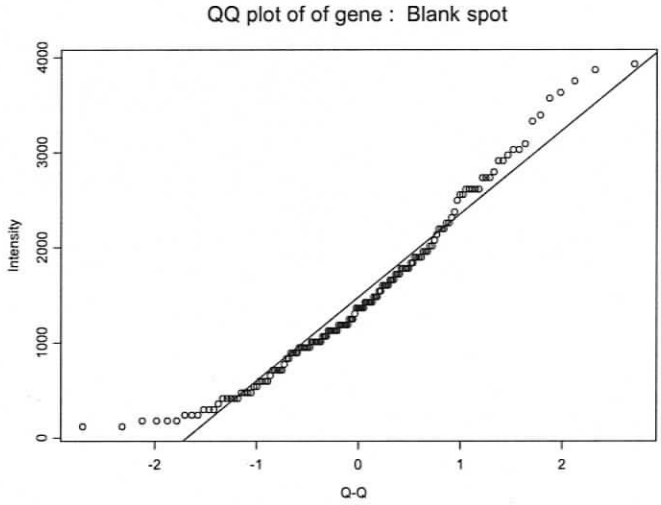

Figure 3-18 Normal QQ plot of pixel intensity for Blank Spot 5

Figure 3-18 shows the QQ plot of pixel intensity of Spot 5. From Table 3-3, it is a good quality spot because all the quality factors are within the pre-set threshold values. However its distribution is somewhat skewed.

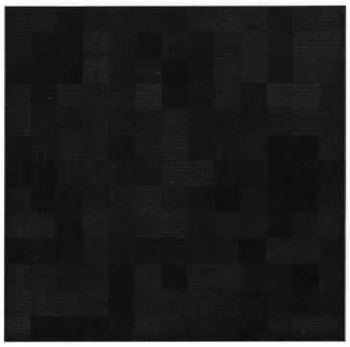

Figure 3-19 Spot image in grey scale for Blank Spot 5

Figure 3-19 is the greyscale image for spot 5.

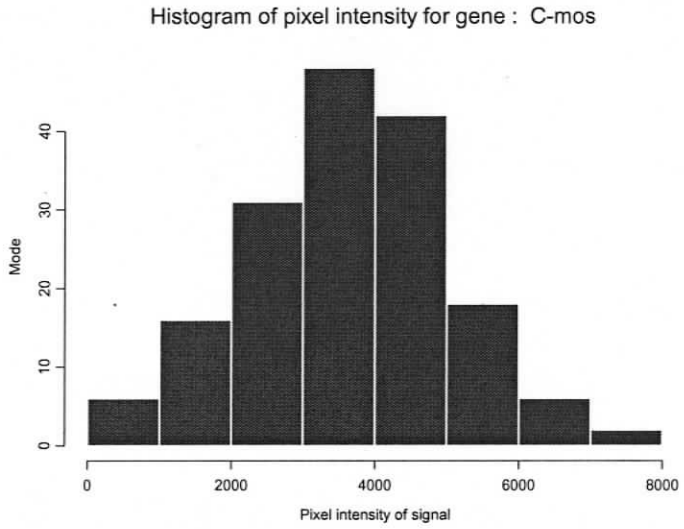

Figure 3-20 Histogram of pixel intensity for Spot 6

Figure 3-20 shows the histogram of pixel intensities for Spot 6, which looks fairly normal.

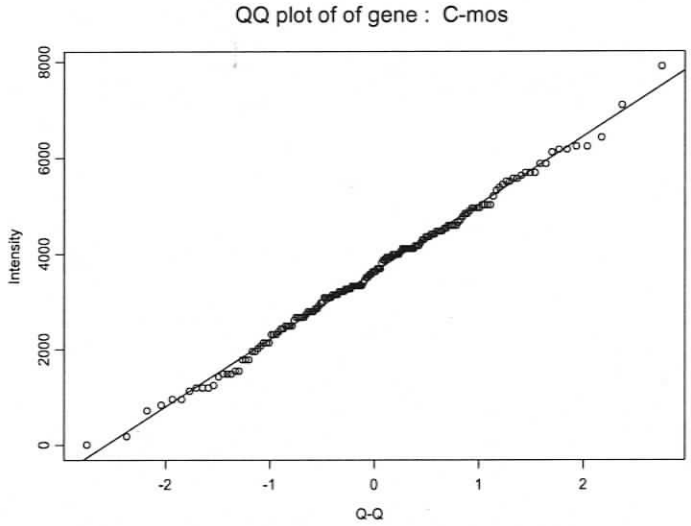

Figure 3-21 Normal QQ plot of pixel intensity for Spot 6

Figure 3-21 shows the QQ plot of pixel intensity of Spot 6. From Table 3-3, it is a good quality spot because all the quality factors are within the pre-set threshold values. It is approximately normally distributed.

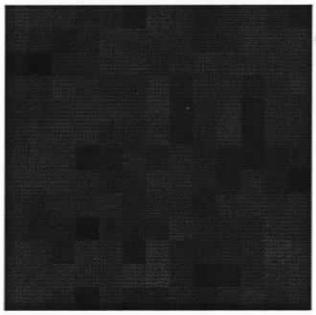

Figure 3-22 Spot image in greyscale for Spot 6

Figure 3-22 shows the greyscale image for Spot 6.

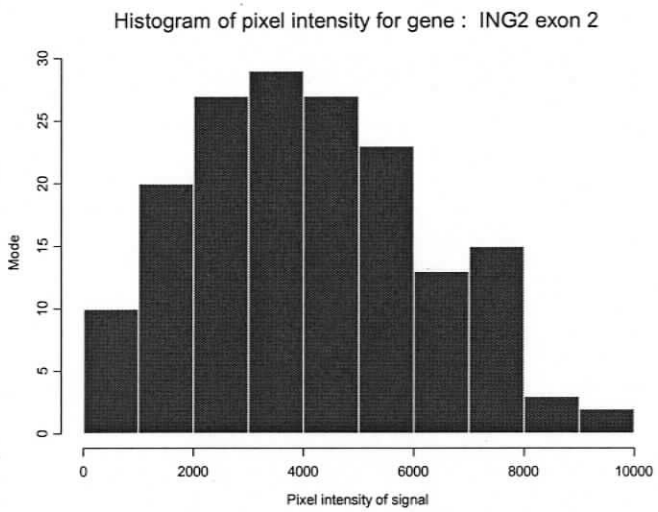

Figure 3-23 Histogram of pixel intensity for Spot 7

Figure 3-23 shows the histogram of pixel intensities for Spot 7. It seems that there are no significant outliers.

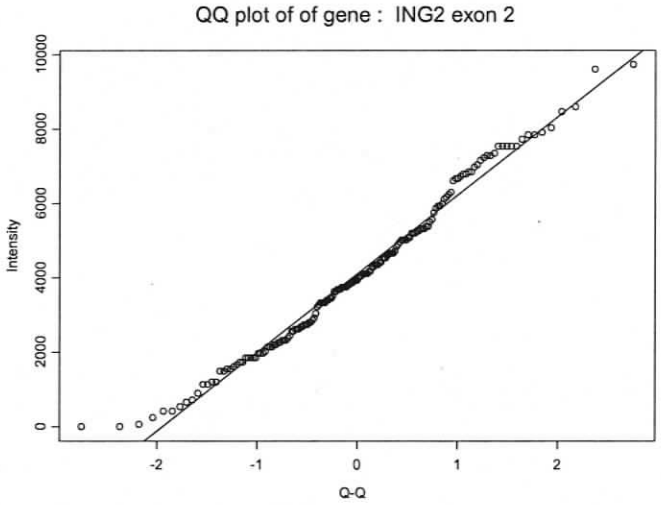

Figure 3-24 Normal QQ plot of pixel intensity for Spot 7

Figure 3-24 shows the distribution of pixel intensity of Spot 7 which is slightly skewed. From Table 3-3, this spot has most of its quality factors satisfying the pre-set

threshold values except the high ignored percentage value (HIP) which is 28.73%, larger than preset value 25%.

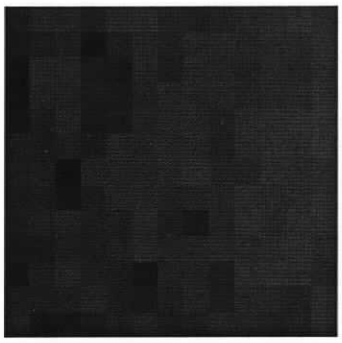

Figure 3-25 Spot image in grey scale for Spot 7

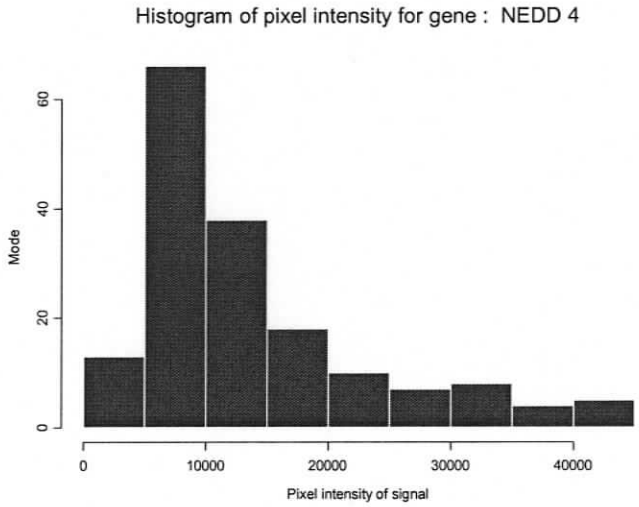

Figure 3-26 Histogram of pixel intensity for Spot 8

Figure 3-26 shows the histogram of pixel intensities for Spot 8. It is very skewed.

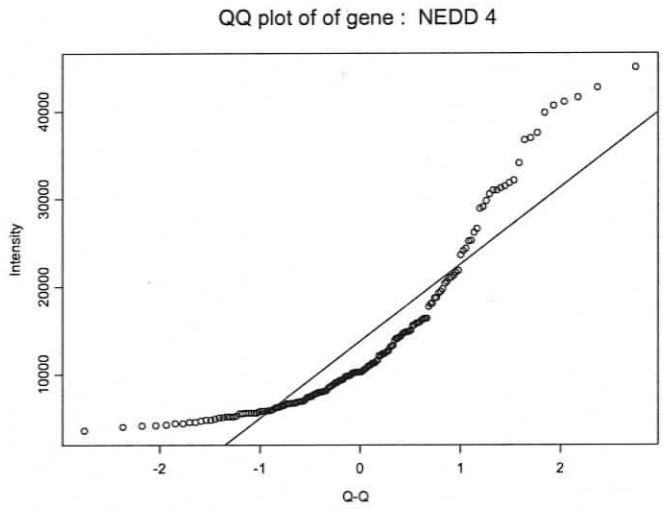

Figure 3-27 Normal QQ plot of pixel intensity for Spot 8

Figure 3-27 shows the distribution of pixel intensity of Spot 8 is obviously not a normal distribution. From Table 3-3, this spot has low shape regularity (ASR) and Area-to-Perimeter (APR) though it has High ignored percentage (HIP) value less than the preset threshold value. Apparently the spot is far away from the expected position. Therefore, this spot is of poor quality.

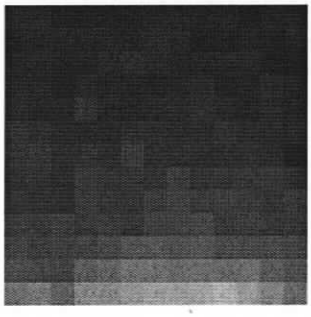

Figure 3-28 Spot image in grey scale for Spot 8

Figure 3-28 shows the spot image for Spot 8 and it shows the irregular shape and it is away from the spot center.

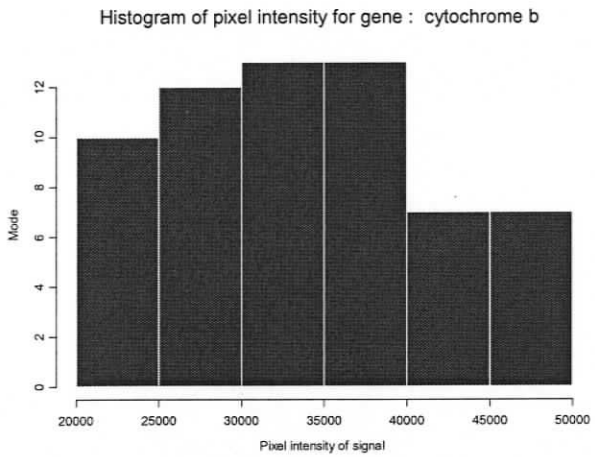

Figure 3-29 Histogram of pixel intensity for Spot 9

Figure 3-29 shows the histogram of pixel intensities for Spot 9. It seems that the pixel intensities are quite high.

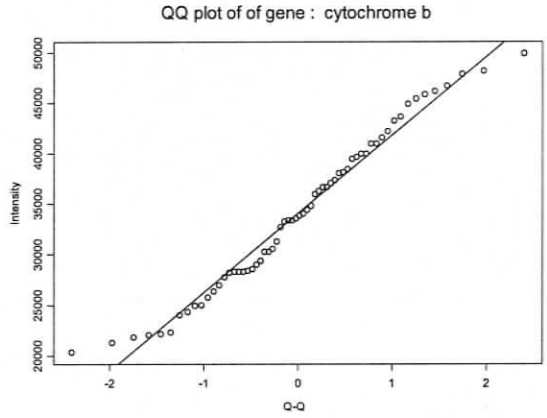

Figure 3-30 Normal QQ plot of pixel intensity for Spot 9

Figure 3-30 shows the QQ plot of pixel intensity for Spot 9. It is approximately normally distributed.

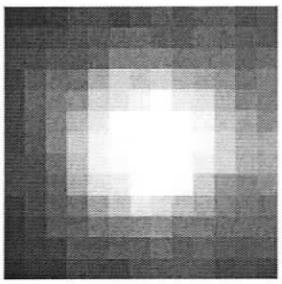

Figure 3-31 Spot image in grey scale for Spot 9

Spot 9 has acceptable values of Area-to-Perimeter (APR), High ignored percentage (HIP) value and Abnormal Shape Regularity (ASR), however it is a saturated spot. Therefore it is considered a poor quality spot.

The above results suggest that just considering one quality factor is not enough. When investigating the quality of a spot, all of these quality factors should be taken into consideration.

3.2 Distributional assumptions for pixel intensities

From the previous section's study on pixel intensity, we find that the distribution of the spot pixel intensities varies considerably. In our analysis, we use the median of the intensity because it is robust. As the sample size grows large, the median is approximately normally distributed. We conducted simulation studies to determine the distribution of the median when the parent distribution is Weibull, Lognormal, Normal and Gamma. In the above examples, the spot area is around 100-200 pixels, so we use a sample size of 125 pixels for the simulation. We simulate 100 samples each of size 125 and obtain the median for each sample. The simulation results are shown below.

For a normal parent distribution, the histogram of the medians and the corresponding QQ plot are shown in Figure 3-32.

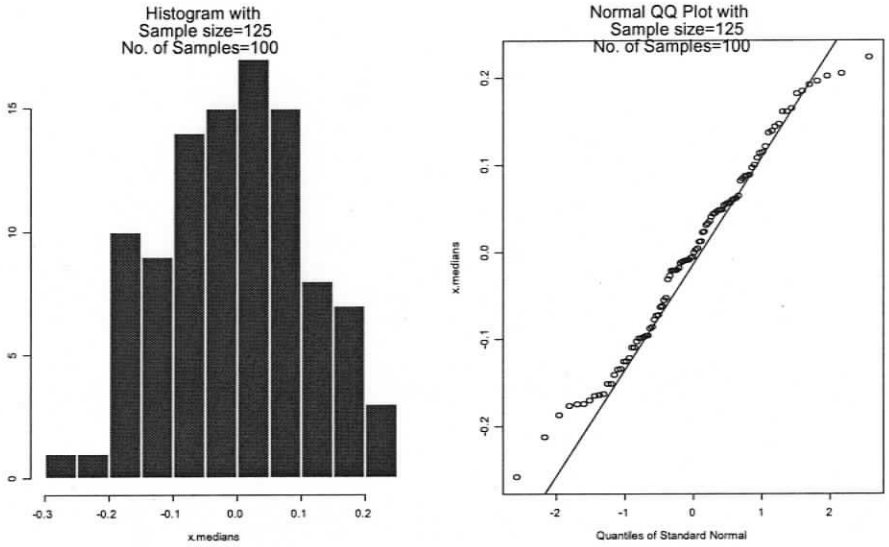

Figure 3-32 Histogram and QQ plot of the median
with Normal parent distribution

For a uniform parent distribution, the histogram of the medians and the corresponding QQ plot are shown in Figure 3-33

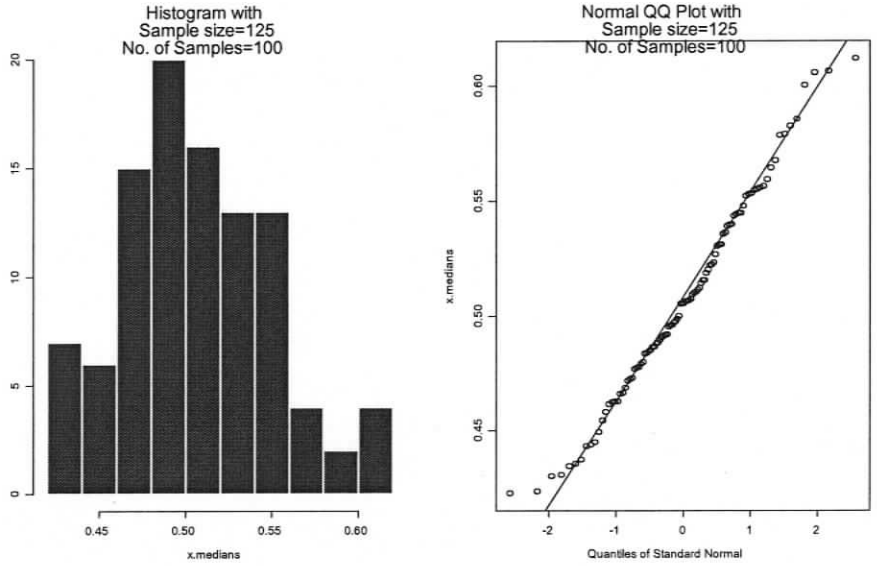

Figure 3-33 Histogram and QQ plot of the median
with Uniform parent distribution

For a Lognormal parent distribution, the histogram of the medians and the corresponding QQ plot are shown in Figure 3-34

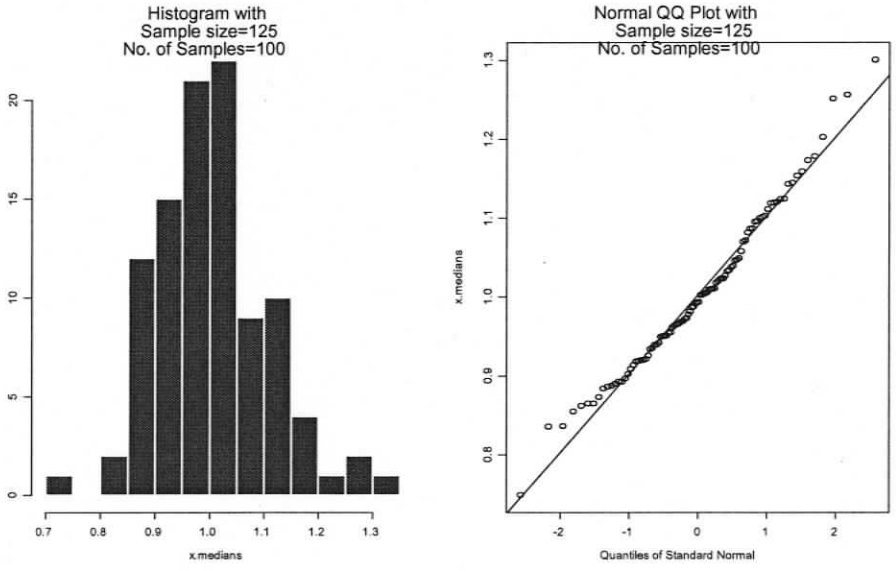

Figure 3-34 Histogram and QQ plot of the median
with Lognormal parent distribution

For a Chi-square parent distribution with 2 degrees of freedom, the histogram of the medians and the corresponding QQ plot are shown in Figure 3-35. The results for other degrees of freedom are very similar to those shown in Figure 3-35.

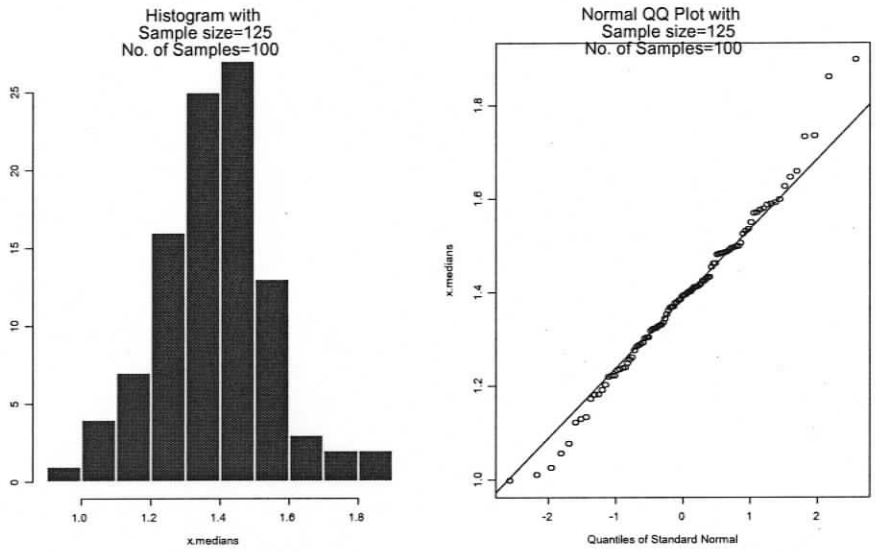

Figure 3-35 Histogram and QQ plot of the median
with Chi-square parent distribution

For a Gamma parent distribution with shape parameter 5, the histogram of the medians and the corresponding QQ plot are shown in Figure 3-36. The parameters of shape and degree of freedom do not make substantial differences.

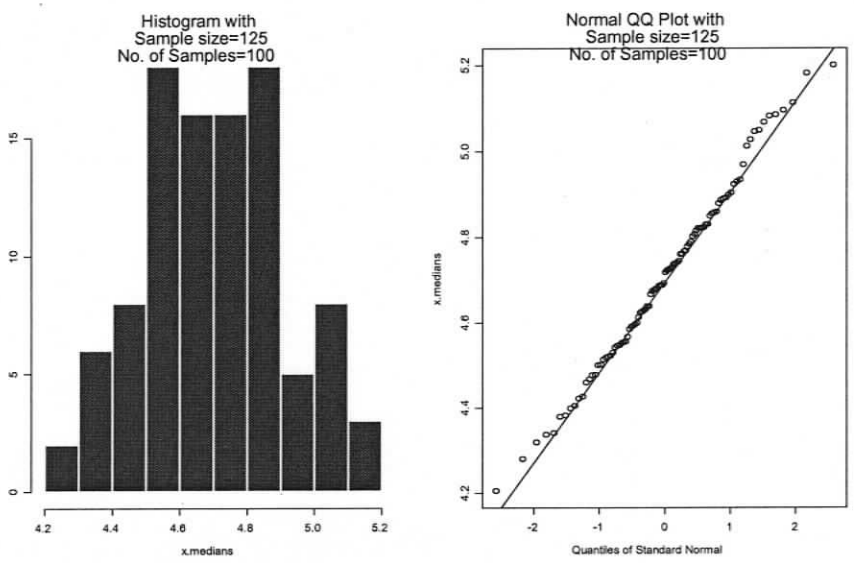

Figure 3-36 Histogram and QQ plot of the median
with Gamma parent distribution

The simulation shows that the distribution of the median is approximately normal for samples of size 125. Therefore it is reasonable to assume that the distribution of the median intensity of each spot is approximately normally distributed.

Chapter 4, Introduction to the statistical model

A new mixture model is proposed which models the effects of background and foreground. Typically there are many blank spots on an array, over 10% of the spots are blank spots on the public and Dr. Helbing's datasets. The blank or empty spots have no cDNA spotted on the array, and we use them to estimate the background intensity distribution.

4.1 Plot to explain the model

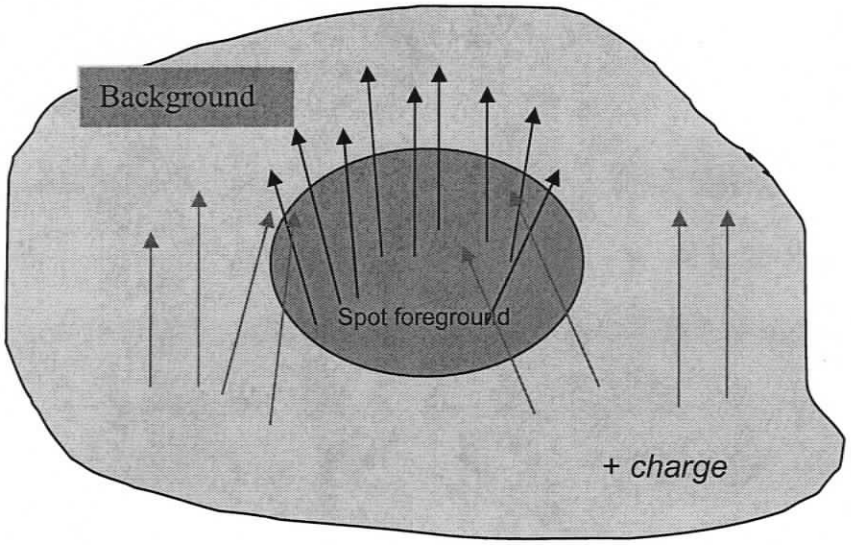

Figure 4-1 Model explanation for blank spot and its background

In Figure 4-1, the dark grey spot represents the signal area and the irregular area around it represents the estimated background. The arrows represent rays of radioactive or fluorescent light emanating up to the scanner from the array. Some rays that emanate from the spot region may be detected as originating from the background region and vice

versa. We call this phenomenon, leakage. If the leakage to the spot is not the same as that to the background, then there will be error associated with both the spot foreground and background estimates.

In practice, the background will be more positively charged than the spot foreground. This is because the array substrate is positively charged to facilitate printing and the cDNA probe negatively charged. Because of the difference in charges, the probe sticks tightly when it touches the array. After printing, the array is washed with a solution that neutralizes the background to some degree, however, it can still be positively charged. The target is negatively charged and therefore there is the possibility of binding to regions other than the spot, in particular, background regions with positive charge. During scanning, the radioactive or florescent material is stimulated and intensities are recorded by the scanning devices. Because the radioactive/florescent rays may scatter, an intensity recorded at a given pixel foreground location may originate from the background and vice versa. Therefore there is leakage from background to foreground and also there is leakage from foreground to background. Our model accommodates this feature of the array.

Because there is no cDNA printed on the blank spots, we use them to estimate the distribution of the background intensity. We investigate the distribution of blank spots after deleting any outliers in the dataset. We could either use the background median or the foreground median of the blank spots in the proposed model. We use the blank spot foreground median to estimate the distribution of background because the foreground of a blank spot has less chance of being affected by neighbouring spots and their

backgrounds. We also investigated the background median in the proposed model and there is no significant difference.

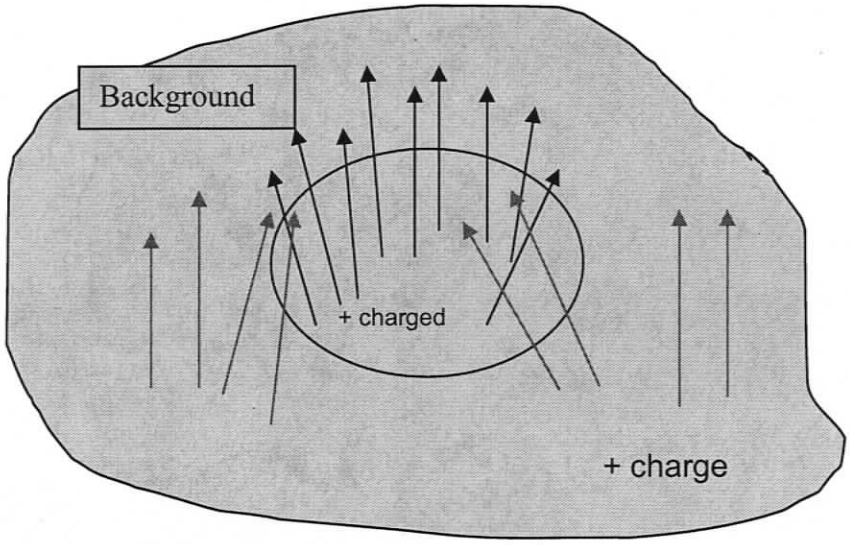

Figure 4-2 Model explanation for a blank spot and its background

Figure 4-2 shows a schematic of a blank spot which has no DNA bound to the spot. Theoretically, the leakage from foreground should be approximately the same as the leakage from the spot regions.

4.2 A Mixture Model

In this section a new model for foreground and background median intensities is proposed. We generically let y_f be the median of the spot/foreground pixel intensities and y_b be the median of the spot background pixel intensities.

For a **blank** spot, let y_f be the median of the spot/foreground pixel intensities. We model

$$y_f = \mu_b + \varepsilon_b$$

where μ_b varies randomly over the slide with unknown distribution, G_0, and ε_b is a mean zero error term.

We estimate G_0 using methods from nonparametric mixture models (Lindsay, 1996). This is totally different from the Bayesian method of Kooperberg et al. (2002). Kooperberg assumes that the prior distribution of background is uniform, an unrealistic assumption in practice. Here we do not have such a restriction. From the discussion in sections 3.1 and 3.2, the median distribution is approximately normal, so the distribution of background is modelled using a normal mixture model

$$y_f \mid \mu_b \sim \mathrm{N}(\mu_b, \sigma^2) \qquad (4\text{-}1)$$

For a **non-blank** spot, let y_b represent the **background** median pixel intensity. We model

$$y_b = \mu_b + S + \varepsilon_b \qquad (4\text{-}2)$$

where again μ_b varies randomly over the slide with unknown distribution, G_0, and ε_b is a mean zero error term. Let S = B – D, the difference between spot/foreground leakage to the background, B, and background leakage to the spot/foreground, D. The background median of a non-blank spot can be affected by the intensity of the spot/foreground. The observed background median is the sum of the true mean background, μ_b, plus the difference between the leakage of foreground and background.

We use the estimate of G_0 from the blank spots in the formulation (4-1). For each spot, we estimate the conditional mean

$$E(\mu_b \mid y_b; G_0, \sigma) = \frac{\int \mu_b \frac{1}{\sigma}\phi(\frac{y_b - \mu_b}{\sigma}) dG_0(\mu_b)}{\int \frac{1}{\sigma}\phi(\frac{y_b - \mu_b}{\sigma}) dG_0(\mu_b)}$$

Substituting estimates for G_0 and σ.

Then S is estimated for each non-blank spot as

$$\hat{S} = y_b - E(\mu_b \mid y_b; \hat{G}_0, \hat{\sigma})$$

The aim is to estimate the mean of signal median for each spot. Let y_f be the median pixel intensities of the spot/foreground for a non-blank spot. We model

$$y_f = \mu_f - S + \varepsilon_f$$

where μ_f varies over the spots on the array, the quantity S is defined above and ε_f is a mean zero error term. Our main interest centers on μ_f which we use to represent the amount of target hybridized to the probe. We estimate μ_f as

$$\hat{\mu}_f = y_f + \hat{S}.$$

4.3 Expected results

It is expected that for the lowly expressed spots, the foreground leakage is less than the background leakage and that for the highly expressed spots, the relationship is the reverse. In other words, for the lowly expressed genes, the spot/foreground leakage is smaller than that of the leakage from background to foreground, therefore B-D<0, that is, S<0. For highly expressed genes, we expect that the spot/foreground leakage is higher than the leakage from the background, so B-D>0, that is, S>0. From the cDNA array

image Figure 1-5, we notice that the brighter spots bleed and affect neighbourhood spots. This is especially true for saturated spots. We expect that S should vary with y_f.

4.4 Mixture models

In the last section, a statistical model is set up to enable background correction and the first step is to estimate the mixing distribution G_0.

4.4.1 What is a mixture model?

Let $Y_i = (Y_{i1},...,Y_{in_i})$ be a vector of random variables arising in the ith cluster (i=1,…, n). Given $(\phi_1,...,\phi_n) \in \Omega^n \subset R^n$, the y_f's are independently distributed with probability density function $h_i(y_i \mid \phi_i)$. Furthermore, suppose that the ϕ_i's are independent random variables with distribution function G, then the Y_i's conform to a mixture model, and are independent with marginal density,

$$f_i(y_i;G) = \int_{\Omega} h_i(y_i \mid \phi) dG(\phi)$$

Here we assume that the mixing distribution G is from the family of all distribution functions defined on Ω.

Mixture models are very useful for the description of data, and there are many areas of application, including fish stock analysis, medicine, economics, bioinformatics, pattern recognition, etc.

When G is a discrete distribution, one can write the model as

$$f(y_i;\pi,\Phi) = \sum \pi_j h(y_i;\xi_j)$$

Where j stands for the j^{th} component, and ϕ takes on values ξ_j. The model is called m component mixture model if j runs from 1 to m. $h(x;\xi_j)$ is the component density function and the ξ_j's are the component parameters. π_j is called component weight or component proportion. The π_j's are non-negative and sum to 1. Φ is a set of possible components that a mixing distribution G that places mass one on ϕ.

4.4.2 Maximum likelihood

The nonparametric maxmum likelihood estimator (NPMLE) theorem is used here to estimate the mixing distribution G_0. We provide a brief introduction to Maximum Likelihood theory below.

The likelihood function is proportional to the joint density function, $f(y_1,...,y_n;\phi)$ of n random variables $Y_1,...,Y_n$ evaluated at $y_1,...,y_n$, and ϕ a scalar or vector parameter. For fixed $y_1,...,y_n$, the likelihood function is a function of ϕ and is often denoted by $L(\phi)$. Usually it is convenient to use the log-likelihood function which is the natural log of the likelihood function, say $\ln(L(\phi))$ and it is often denoted by $l(\phi)$

The maximum likelihood estimator (MLE) is the estimator which maximizes the likelihood function. That is, for a given set of observation S($y_1,...,y_n$), a value $\hat{\phi}$ in parameter space Ω at which $L(\phi)$ is a maximum. As the logarithm function is increasing, $\hat{\phi}$ also maximizes the log-likelihood function. So in practice the log-likelihood is often used to find the MLE.

Why should we use an MLE? The MLE is an important type of estimator for the following reasons (Bain, 1992).

- MLEs are often simple and easily computed;
- MLEs have asymptotic optimality properties, consistency and efficiency;
- MLEs are invariant under reparameterization;
- If an efficient estimator exists, it will be the MLE.

4.4.3 Optimization problem

Here we use nonparametric Maximum Likelihood (NPML) theory to obtain the MLE of G_0 (Lindsay, 1996). Here we assume that $Y_1,...,Y_n$ are independent with $L_i(G) \propto f_i(y_i;G)$, the marginal density, and $L_i(\phi) \propto h_i(y_i \mid \phi)$, the conditional density of Y given ϕ . The likelihood function of the mixture model can be represented as

$$L(G) = \prod_{i=1}^{n} L_i(G) = \prod_{j=1}^{D} [\int L_j(\phi) dG(\phi)]^{n_j}$$

where D is the number of distinct observations and n_j is the multiplicity of the j^{th} distinct observation.

To maximize the above objective function, it is convenient to consider the log-likelihood function, that is

$$l(G) = \ln(L(G)) = \sum_{j}^{D} n_j \ln(\int L_j(\phi) dG(\phi)) \qquad (4\text{-}3)$$

Note that the parameter G is from an infinite-dimensional space, and we require the maximizing distribution from that space. To determine the MLE, we use a quantity that is quite similar to a derivative called a directional derivative. It is defined as

$$D(L_{G_1}; L_{G_0}) = \lim_{\varepsilon \to 0} \frac{l\{(1-\varepsilon)G_0 + \varepsilon G_1\} - l(G_0)}{\varepsilon}$$

When the distribution $G_1 = \Delta_\phi$ is a point mass distribution with mass at a single point ϕ, the directional derivative can be written as

$$D_{G_0}(\phi) = \sum [\frac{L_i(\phi)}{L_i(G_0)} - 1] \qquad (4\text{-}4)$$

To solve the optimization problem we use the following framework:

- The feasible region can be formulated using vectors of likelihood components. Let $\Gamma = \{\Psi(\phi) : \phi \in \Omega\}$, where $\Psi(\phi) = [L_1(\phi),...,L_D(\phi)]^T$ which is a vector of likelihood components. Then define M=$\{\ \Psi(G) = (L_1(G),...,L_D(G))^T\}$, where $L_j(G) = \int L_j(\phi) dG(\phi)$ and G is a probability measure. The set M is the convex hull of Γ, i.e. M =conv(Γ). M contains the feasible vectors of likelihood components over all distributions G.
- The log-likelihood function (4-3) as the objective function is just a function of the vector of likelihood components and is used as the objective function.
- When getting the maximum $\hat{L} = L(\hat{G})$, construct the $\hat{G}$, with the property that for any other Δ_ϕ, the directive derivatives from $\hat{G}$ to any direction Δ_ϕ is less than or equal to 0. This process is just like climbing up a mountain. Each time you arrive at a location, you look around for the next location. If there is a location towards which the directive derivative is greater than 0, then move to that location. This will move you up the mountain. If the directional derivative in any direction is equal to 0 or negative, then you have already reached the summit.

The existence and structural properties of NPMLE are proved in Lindsay (1996). There are four basic theorems about the NPMLE from Lindsay (1996, pp. 115-6).

Theorem 1: Suppose that Γ is closed and bounded and that M contains at least one point with positive likelihood. Then there exists a unique $\hat{L} \in \partial M$, which is the boundary of M, such that $\hat{L}$ maximizes the log-likelihood over M.

Theorem 2: The following three statements are equivalent:

- $\hat{G}$ maximizes $l(G) = \ln(\mathrm{L(G)})$ over $\mathrm{G} \in \mathrm{M}$;
- $\hat{G}$ minimizes $\sup_{\phi} D_G(\phi)$;
- $\sup_{\phi} D_{\hat{G}}(\phi) = 0$, for $\phi \in \Omega$.

Theorem 3: The support of any maximum likelihood estimator $\hat{G}$ lies in the set $\{\phi : D_{\hat{G}}(\phi) = 0\}$

Theorem 4: The solution $\hat{L}$ can be represented as $L(\hat{G})$, where $\hat{G}$ has no more than D points of support.

4.5 Algorithm Introduction

In practice, there are several algorithms used to estimate the NPMLE. These algorithms can be grouped into three categories:

- Based on the expectation maximum (EM) algorithm;
- Based on the directional derivative.

Lesperance & Kalbfleisch (1992) demonstrate how semi-infinite programming (SIP) methodology can be used to obtain the NPMLE. They use an algorithm due to Coope & Watson (1985) to solve the corresponding semi-infinite programming problem.

There are many variations of the EM algorithm. It has some advantages: it is easy to program, theoretically guaranteed to converge and reliable and economy of storage. However, it has some disadvantages such as it is a local maximum seeker and it is sensitive to the initial value. Other shortcomings include: the convergence speed is linear which is very slow if the problem size is large, and one must provide the number of components at the start because it does not determine the number of components in the mixture. The latter is a big problem since in many cases the number of components is unknown.

Three algorithms based on the directive derivative are: the vertex direction method (VDM), Wu (1978), its modification as the vertex exchange method (VEM), Bohning (1986) and the intra-simplex direction method (ISDM), Lesperance & Kalbfleisch (1992).

The three algorithms based on the directional derivative provide global convergence. For both VDM and VEM, the NPMLE can be obtained from any initial discrete probability distribution. However the convergence speed is very slow, though VEM is faster than VDM and VEM can even discard "bad" components and add "good" components. What is more, both algorithms have to keep track of and perform computations over all of the support points at every iteration.

Lesperance and Kalbfleisch (1992) proposed the Intra-simplex Direction Method (ISDM) which is a much more reliable and efficient algorithm, compared with the other methods introduced above. ISDM is described as follows.

1) Set the initial estimate G_1 and j=1.

2) Compute all local maxima $\phi_{j1}^*,...,\phi_{jm_j}^*$ of $g(\phi, G_j), \phi \in \Omega$, such that $g(\phi_{js}^*, G_j) \geq 0 \quad (s = 1,..., m_j < k)$, where $g(\phi, G_j) = D_{G_j}(\phi)$ is the directional derivative computed using formulation (4-4).

3) If $\max_s g(\phi_{js}^*, G_j) = 0$ stop

4) Compute $\varepsilon_{j0},..., \varepsilon_{jm_j}$, the values that maximize

$$l\{\varepsilon_0 G_j + \sum_{s=1}^{m_j} \varepsilon_s \Delta_{\phi_{js}^*}\} = \sum_r n_r \log\{\varepsilon_0 L_r(G_j) + \sum_{s=1}^{m_j} \varepsilon_s L_r(\phi_{js}^*)\}$$

subject to $\sum_{s=0}^{m_j} \varepsilon_s = 1$ and $\varepsilon_s \geq 0 \quad (s = 0,1,..., m_j)$.

5) $G_{j+1} = \varepsilon_{j0} G_j + \sum_{s=1}^{m_j} \varepsilon_{js} (\Delta_{\phi_{js}^*})$.

6) Set j=j+1 and go to 2.

ISDM is considered to be a reliable and fast algorithm to find the global maximum. The original implementation used the NAG subroutine E04VDF which solves the optimization problem in step 4) subject to bounded and linear constraints.

To make the implementation more flexible so that it can be used by PC users with S-Plus and R, we chose to use another well tested subroutine to solve the problem. www.netlib.org provides a set of subroutines to solve the minimization problem with bound constraints. Because in our NPMLE there is a linear constraint, we used a Lagrange multiplier in step 4) in the algorithm above. That is:

4') $l\{\varepsilon_0 G_j + \sum_{s=1}^{m_j} \varepsilon_s \Delta_{\theta_{js}^*}\} = \sum_r n_r \log\{\varepsilon_0 L_r(G_j) + \sum_{s=1}^{m_j} \varepsilon_s L_r(\theta_{js}^*)\} + \lambda(\sum_{i=0}^{m} \varepsilon_i - 1)^2$

subject to $\varepsilon_s \geq 0 \quad (s = 0,1,...,m_j)$

In the implementation, the initial selection of Lagrangian multiplier must be chosen with care. Because the number of spots is large, then the first term of the sum in above formula can be very large. If the Lagrangian multiplier is too small compared with the first term of the sum above, then the global maximum of the objective function cannot be reached. Here I take the range of the medians of the pixel intensities into account. If the range is 1000, I take the initial value to be $10^7 = 10^{3+4}$. If the number of digits in the range is k, then the initial value is 10^{3+k}. This setting was developed from computing experience using different datasets.

Another problem encountered in the implementation is overflow, because there are many exponential operations and there are upper and lower limits for the exponential operation. If the values are too small (negative), then the value of the exponential will be 0. If this result is used in a denominator, then problems will arise. In the program, if this happens, a large constant is used to scale the term where the small amount will appear and then relative changes are made to the expression to maintain the correct value. The software keeps careful track of the operations to solve this problem.

Chapter 5, Application of mixture model in background correction

5.1 More details on Helbing Lab's data

The datasets from Dr. Caren Helbing's lab consist a group of experiments referred to as EPA Study B, Brain T3 series (Helbing et al. 2006). The *Xenopus laevis* tadpole exposures were conducted by the EPA and the samples were then transferred to them for processing and analysis by array and QPCR.

There are in total six treatments: one is a control and 5 different concentrations of T3 0.31ppb, T3 0.63ppb, T3 1.25ppb, T3 2.5ppb and T3 5.0ppb. Three time points are taken into account: 24 hours, 48 hours and 96 hours. There are replicate arrays representing 3 individual animals per treatment time-point combination (except T3 1.25ppb 96h where there are only 2 replicates). In total, there are 53 array datasets.

The datasets are tab-delimited text output files from ImaGene, both from non-auto levelled images ("na") and from auto-levelled images ("a"), and excel files. The auto-levelled datasets are not pure raw data. In this thesis, we use the non-autolevelled files.

All these raw data are obtained by ImaGene with only the indicators of "background/foreground being contaminated" set to be true, while other quality factors are not turned on. In my analysis, before fitting the model, I applied data cleaning on the raw dataset using the criteria listed in Table 5-1.

Table 5-1 Quality factor setting for data analysis

No	Quality factor setting	Quality factor	Variable in software	Critical value
1	Shape Regularity	ASR	shapeReg	<0.65>
2	Ignored Area	HIP	ignoredArea	<0.25>
3	Area to Perimeter ratio	APR	area2Per	<0.65>
4	Open Perimeter ratio	OPP	openPer	<0.3>
5	Background contaminated flag	BC	bgdcFlag	<1>
6	Signal contaminated flag	FC	fgdcFlag	<1>
7	Saturated spot value		-	<50000>

The definition and description of the criteria in Table 5-1 were introduced in Chapter 2, Section 2.3. Before G_0 was computed, spots which failed any of the criteria were removed from the dataset.

5.2 Discussion on outliers

Even after data cleaning, there are still unusual summary statistics in the dataset because quality control screens are not applied on the pixel level. For example, even though a spot is not considered a saturated spot, there may be some saturated pixels within that spot which cause the summary statistics for that spot to be unusual. In our analysis, the outliers in blank spots are removed before fitting the model and estimating the distribution of the background.

Figure 5-1 contains a box or whisker plot. The plot provides a display of a five-number data summary: median, upper and lower quartiles, the minimum and maximum data values. The box itself contains the middle 50% of the data. The upper edge of the box indicates the 75^{Th} percentile of the data while the lower edge of the box indicates the 25^{th} percentile. The range of their difference is known as the inter-quartile range. The

horizontal line in the box indicates the median of the data. Usually, the ends of the vertical lines or "whiskers" indicate the minimum and maximum of the data. If there are outliers, the line is extended to 1.5 times the inter-quartile range (IQR). Any point outside the ends is considered to be an outlier or suspected outlier. Figure 5-1 is from http://www.netmba.com/statistics/plot/box/.

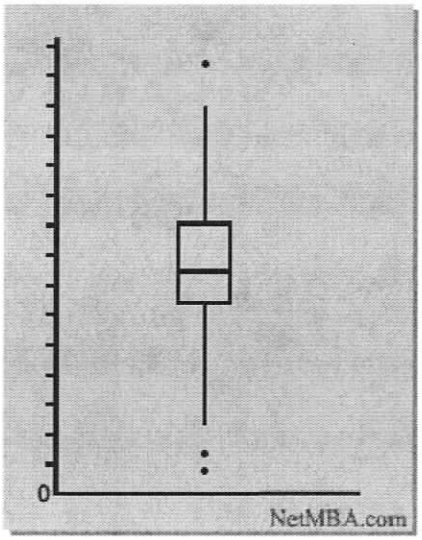

Figure 5-1 Box-Plot

Before applying the mixture model on blank spots to estimate the background distribution, outliers as defined above are removed from the background medians. As there are no cDNA on the blank spots, then the difference between the background and foreground medians should have mean zero with an approximate normal distribution if our model is appropriate. The histogram (Figure 5-2) and the normal QQ plot (Figure 5-3) suggest that the differences are approximately normally distributed with mean zero. The data for Figures 5-2 and 5-3 comes from the dataset T3 0.31ppb 24hour, sample 14.

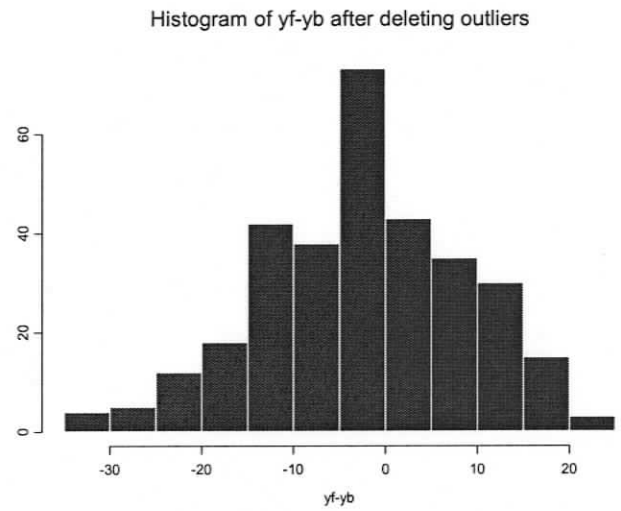

Figure 5-2 Histogram for the difference between background and foreground median

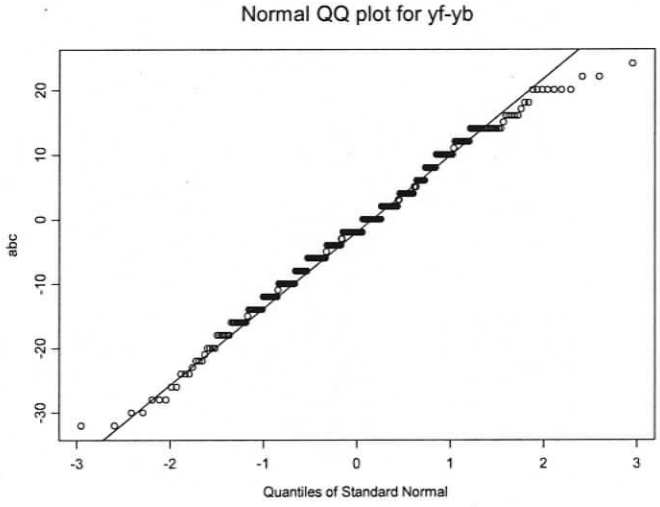

Figure 5-3 Normal QQ plot for the difference between background and foreground median

5.3 Applying model to the dataset

Here only the results for T3 0.31ppb at three time points are presented. The results from the other datasets are quite similar. The datasets used here are T3 0.31ppb 24h#14,

T3 0.31ppb 48h#17 and T3 0.31ppb 96h#22. Table 5-2 is a list of the figures associated with these three data sets.

Table 5-2 Figure index for the result of case study

No	Dataset	Figure Type	Figure No
1	T3 0.31ppb 24h#14	Histogram of difference	Fig 5-4
2	T3 0.31ppb 24h#14	Normal QQ plot of difference	Fig 5-5
3	T3 0.31ppb 24h#14	Histogram of $\hat{S}$	Fig 5-6
4	T3 0.31ppb 24h#14	Plot of $\hat{S}$ versus y_f	Fig 5-7
5	T3 0.31ppb 24h#14	Plot of $\hat{\mu}_f$ versus $y_f - y_b$	Fig 5-8
6	T3 0.31ppb 24h#14	Plot of $\hat{\mu}_f$ versus y_f	Fig 5-9
7	T3 0.31ppb 24h#14	Plot of E($\hat{\mu}_b \mid y_b; \hat{G}_0, \hat{\sigma}$) versus y_b	Fig 5-10
8	T3 0.31ppb 48h#17	Histogram of difference	Fig 5-11
9	T3 0.31ppb 48h#17	Normal QQ plot of difference	Fig 5-12
10	T3 0.31ppb 48h#17	Histogram of $\hat{S}$	Fig 5-13
11	T3 0.31ppb 48h#17	Plot of $\hat{S}$ versus y_f	Fig 5-14
12	T3 0.31ppb 48h#17	Plot of $\hat{\mu}_f$ versus $y_f - y_b$	Fig 5-15
13	T3 0.31ppb 48h#17	Plot of $\hat{\mu}_f$ versus y_f	Fig 5-16
14	T3 0.31ppb 48h#17	Plot of E($\hat{\mu}_b \mid y_b; \hat{G}_0, \hat{\sigma}$) versus y_b	Fig 5-17
15	T3 0.31ppb 96h#22	Histogram of difference	Fig 5-18
16	T3 0.31ppb 96h#22	Normal QQ plot of difference	Fig 5-19
17	T3 0.31ppb 96h#22	Histogram of $\hat{S}$	Fig 5-20
18	T3 0.31ppb 96h#22	Plot of $\hat{S}$ versus y_f	Fig 5-21
19	T3 0.31ppb 96h#22	Plot of $\hat{\mu}_f$ versus $y_f - y_b$	Fig 5-22
20	T3 0.31ppb 96h#22	Plot of $\hat{\mu}_f$ versus y_f	Fig 5-23
21	T3 0.31ppb 96h#22	Plot of E($\hat{\mu}_b \mid y_b; \hat{G}_0, \hat{\sigma}$) versus y_b	Fig 5-24

Notes:

- Histogram of difference: Shows the difference between background and foreground median of the **blank** spots

- Normal QQ plot: Shows whether the difference between background and foreground median for the **blank** spots follows a Normal distribution
- Histogram of $\hat{S}$: Shows the estimate of difference of leakage between foreground and background
- Plot of $\hat{S}$ versus y_f : Shows the relationship of $\hat{S}$ to y_f for **nonblank** spots
- Plot of $\hat{\mu}_f$ versus $y_f - y_b$: Shows the relationship of $\hat{\mu}_f$ to $y_f - y_b$ for **nonblank** spots
- Plot of $\hat{\mu}_f$ versus y_f : Shows the relationship of the estimated spot expression value to y_f for **nonblank** spots
- Plot of $E(\hat{\mu}_b \mid y_b; \hat{G}_0, \hat{\sigma})$ versus y_b : Shows the relationship of the mean of the posterior background expression to y_b for **blank** spots.

5.3.1 Results for T3 0.31ppb 24h#14

- Histogram and normal QQ plot of the $y_f - y_b$, the difference between the median of foreground and background for 318 blank spots. There are 325 blank spots in total and 7 spots are removed because their ($y_f - y_b$)'s are outside the interval ($Q_{.25} - 1.5 * IQR$, $Q_{.75} + 1.5 * IQR$) where $Q_{.25}$ and $Q_{.75}$ are the first and third quantiles of $y_f - y_b$ for the blank spots .

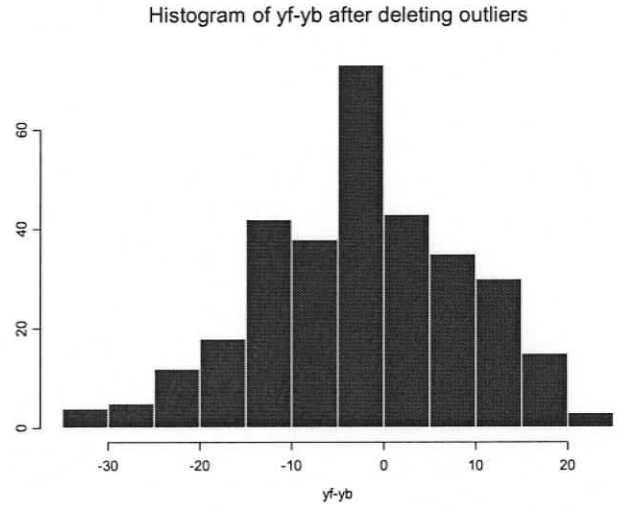

Figure 5-4 Histogram of difference between
Background and foreground median

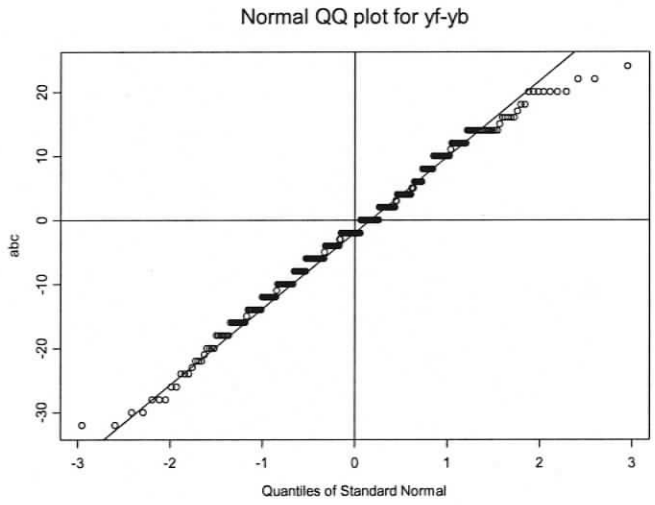

Figure 5-5 Normal QQ plot of difference
between background and foreground median

Figure 5-4 and Figure 5-5 suggest that the difference between background and foreground median for blank spots follows a normal distribution with mean zero. This satisfies the model assumptions and conditions. The median of the

blank spot foreground is then used to estimate the distribution of the background intensity distribution.

- Estimate of G_0, which is the distribution of background pixel intensity using the software provided in this thesis. The maximum of the parameter ϕ is greater than 0 and less than the maximum of y_f for the blank spot which is 358.

The number of spots is 300
The range of the ϕ is [0, 358]
$l(\hat{G}_0)$: -1347.2306177536
Num of supports m = 10

ϕ	π	$D_G(\phi)$
256.356957611924	0.0898317736776989	-0.00598637232372956
263.518785804712	0.0124603022426456	-0.00652955967729718
272.360316870647	0.24183275887469	-0.0061087018048519
284.24516303362	0.218879332696918	-0.00565322711923644
297.148965818508	0.196571410481293	-0.00569520503966769
307.483115105823	0.0906132647519072	-0.00580852641083807
323.399824759093	0.0872551957583336	-0.00511065614870598
330.758540407969	0.0224458377476581	-0.00612534170564993
340.900155072932	0.0137959708404879	-0.00597018814440253
350.76008391576	0.0264641181535763	-0.00611487235993557

From the above result, it shows that the distribution of G_0 can be described as mixture model of 13 normal distributions with respective means ϕ 's and proportion π 's.

- Histogram of $\hat{S}$

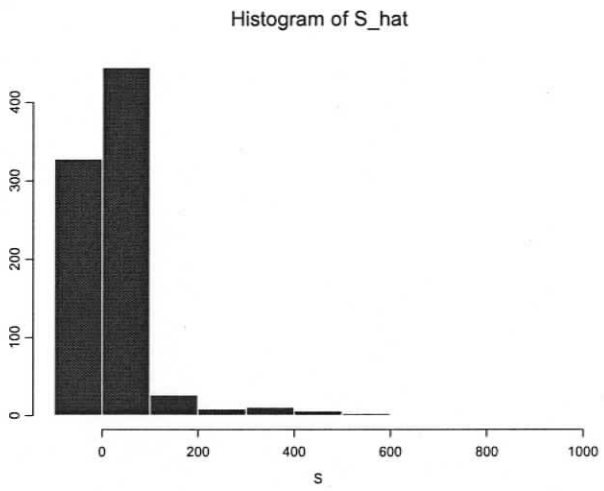

Figure 5-6 Histogram of estimate $\hat{S}$

Figure 5-6 shows the estimate of the leakage between foreground and background. Note that there are negative values. The following is the statistical summary of estimate of $\hat{S}$.

- Summary of $\hat{S}$

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-20.6751	-1.7312	1.1529	21.6070	3.1961	905.8552

- Plot of $\hat{S}$ versus y_f

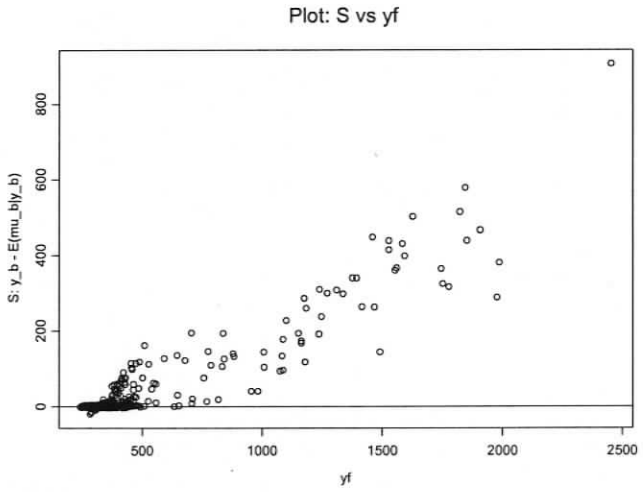

Figure 5-7 Plot of $\hat{S}$ versus y_f

Figure 5-7 shows the relationship of the leakage between spot/foreground and background and the pixel intensity of the spot/foreground. It seems in some lowly expressed genes, the spot/foreground leakage is smaller than that of the leakage from background to foreground, therefore there are some $\hat{S} < 0$. As y_f increases, the leakage of spot/foreground is larger than that of the leakage from background to foreground. $\hat{S}$ tends to increase with y_f.

- Plot of $\hat{\mu}_f$ versus $y_f - y_b$

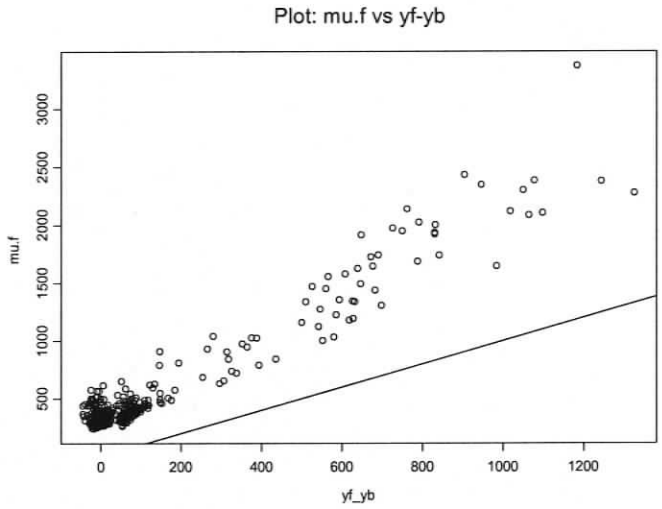

Figure 5-8 Plot of $\hat{\mu}_f$ versus $y_f - y_b$

Figure 5-8 show the relationship between $\hat{\mu}_f$ and the difference of median of the pixel intensity between foreground and background, ie. $y_f - y_b$. It shows that as $y_f - y_b$ increases, so does $\hat{\mu}_f$.

- Plot of $\hat{\mu}_f$ versus y_f

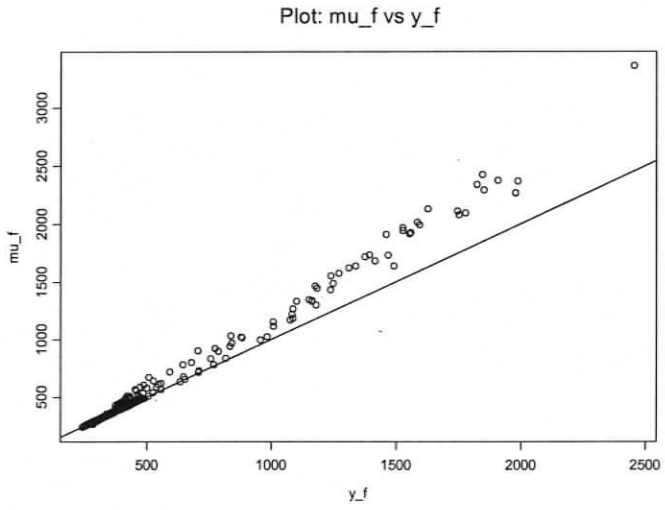

Figure 5-9 Plot of $\hat{\mu}_f$ versus y_f

Figure 5-9 shows the relationship between the mean of the corrected foreground pixel intensity and the median of foreground pixel intensity which is the uncorrected value. As it can be seen that the leakage from the foreground to background is larger than the leakage from background to foreground as y_f increases, therefore, after background correction, as y_f increases, the "true" foreground pixel intensity is larger than the uncorrected value.

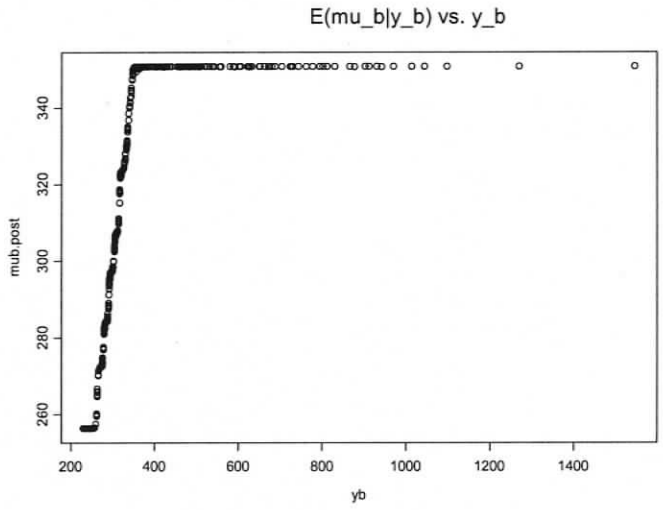

Figure 5-10 Plot of $\mathbf{E}(\hat{\mu}_b \mid y_b; \hat{G}_0, \hat{\sigma})$ versus y_b

Figure 5-10 shows the mean of the posterior of the background pixel intensities versus the background median.

From the above plots we can see that there exists the relationship between the estimated gene expression value $\hat{\mu}_f$, and y_f. It can be noticed that when $\hat{\mu}_f$ is

small, then the background might have more leakage to foreground, and as $\hat{\mu}_f$ becomes larger, the leakage from foreground becomes larger.

5.3.2 Results for T3 0.31ppb 48h#17

- Histogram and normal QQ plot of the $y_f - y_b$, the difference between the median of foreground and background for 314 blank spots. There are 326 blank spots before the outlier deletion.

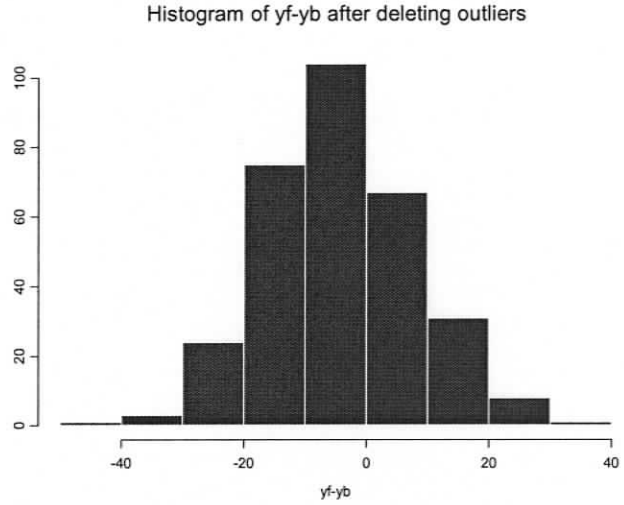

Figure 5-11 Histogram of difference between Background and foreground median

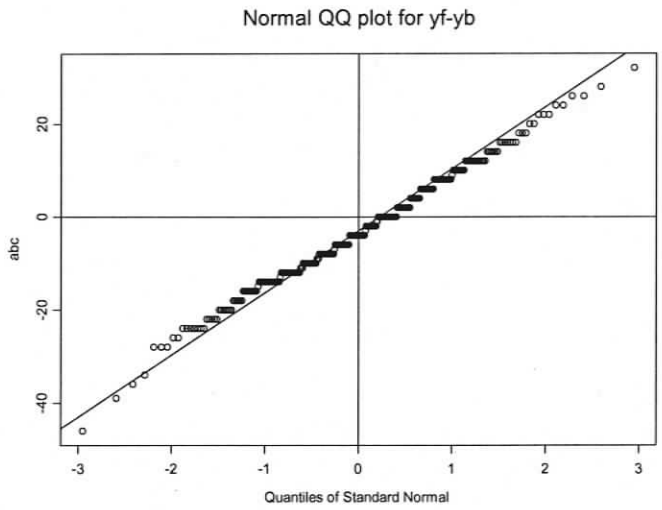

Figure 5-12 Normal QQ plot of difference
of background and foreground median

Figure 5-11 and Figure 5-12 suggest that the difference between background and foreground median for blank spots follows normal distribution with mean zero. This satisfies the model assumptions and conditions.

- Estimate of G_0 which is the distribution of background. The maximum of the parameter ϕ is greater than 0 and less than the maximum of y_f for the blank spot which is 462.

The number of spots is 314
The range of the ϕ is [0, 462]
$l(\hat{G}_0)$: -1435.27316714925
Num of supports m = 14

ϕ	π	$D_G(\phi)$
307.02069592545	0.00640728257542057	-0.0109808806122293
329.321524789727	0.0197097978658725	-0.00292049432064534
342.510676294528	0.0978821751865766	-0.00134073630817189
353.306954740905	0.19252301210876	-0.00174993175670912
362.636925864428	0.18556568596017	-0.00257066589219601
371.787759493984	0.142208177254862	-0.00234251359648496
382.426185471949	0.116279726813819	-0.00321347382895043
394.875829841894	0.091782733161403	-0.00211845196977045

404.118816067621 0.0346459061180894 -0.00266288407111581
416.667542700078 0.0613229427940994 -0.00240642116936218
426.803453342842 0.0259699752798261 -0.0026470780231298
435.383275570211 0.00971437069887498 -0.00450426996299758
446.112927805677 0.0127539527765585 -0.00332540141997484
461.849131582634 0.00324968849126334 -0.0181966511896743

From the above result, it shows that the distribution of G_0 can be described as mixture model of 14 normal distributions with respective means ϕ's and proportion π's.

- Histogram of $\hat{S}$

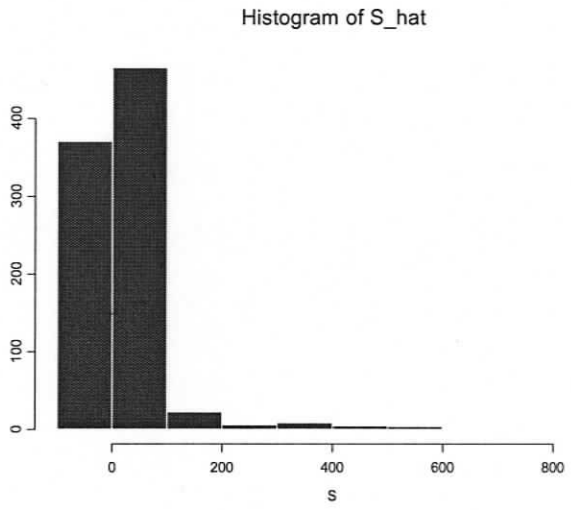

Figure 5-13 Histogram of estimate $\hat{S}$

Figure 5-13 shows the estimate of the leakage between foreground and background. Note that there are negative values. The following is the statistical summary of estimate of $\hat{S}$.

- Summary of $\hat{S}$

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-5.3215	-1.4100	0.6311	17.7740	2.5797	822.1509

- Plot of $\hat{S}$ versus y_f

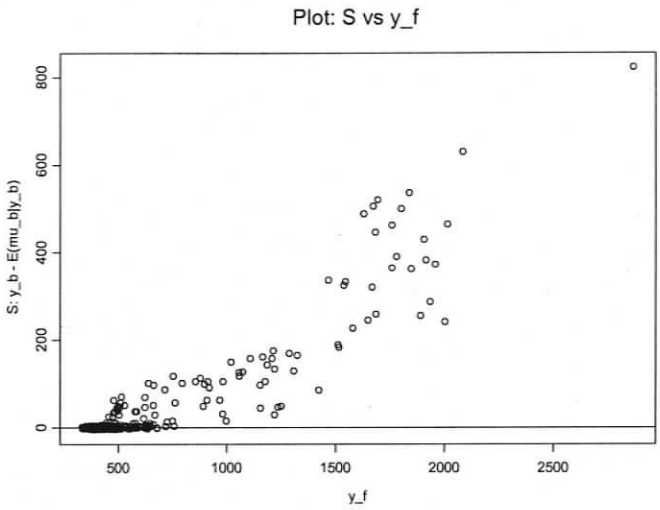

Figure 5-14 Plot of $\hat{S}$ versus y_f

Figure 5-14 shows the relationship of the leakage between spot/foreground and background and the median of pixel intensity of the spot/foreground. It seems in some lowly expressed genes, the spot/foreground leakage is smaller than that of the leakage from background to foreground, therefore there are some $\hat{S} < 0$. As y_f increases, the leakage of spot/foreground is larger than that of the leakage from background to foreground. $\hat{S}$ tends to increase with y_f.

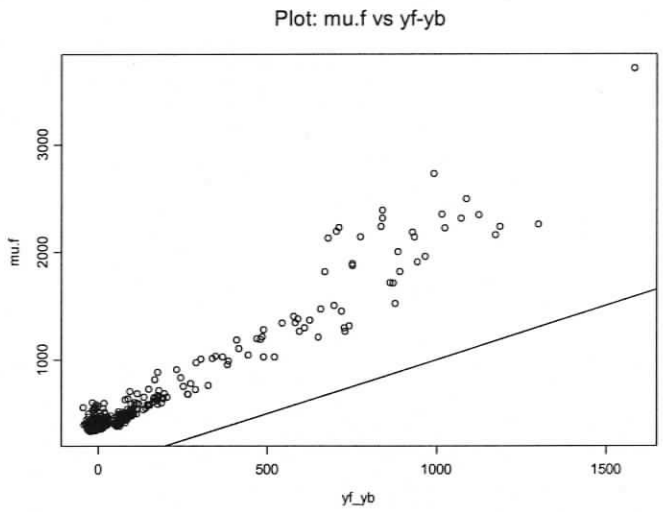

Figure 5-15 Plot of $\hat{\mu}_f$ versus $y_f - y_b$

Figure 5-15 show the relationship between $\hat{\mu}_f$ and the difference of median of the pixel intensity between foreground and background, ie. $y_f - y_b$. It shows that as $y_f - y_b$ increases, so does $\hat{\mu}_f$.

- Plot of $\hat{\mu}_f$ versus y_f

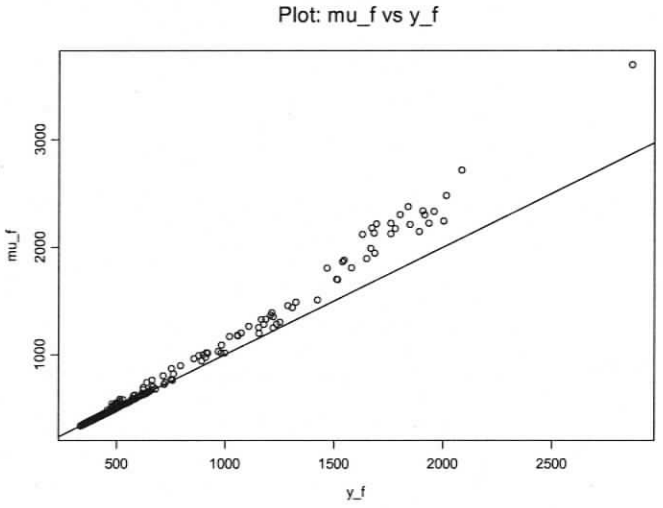

Figure 5-16 Plot of $\hat{\mu}_f$ versus y_f

Figure 5-16 shows the relationship between the mean of the corrected foreground pixel intensity and the median of foreground pixel intensity which is the uncorrected value. As it can be seen that the leakage from the foreground to background is larger than the leakage from background to foreground as y_f increases, therefore, after background correction, as y_f increases, the "true" foreground pixel intensity is larger than the uncorrected value.

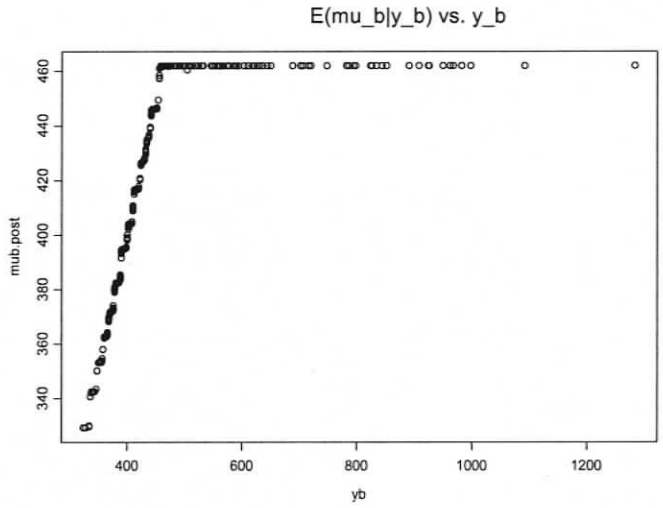

Figure 5-17 Plot of $\mathbf{E}(\hat{\mu}_b \mid y_b; \hat{G}_0, \hat{\sigma})$ versus y_b

Figure 5-17 shows the mean of the posterior of the background pixel intensities versus the background median.

5.3.3 Results for T3 0.31ppb 96h#22

- Histogram and normal QQ plot of the $y_f - y_b$, the difference between the median of foreground and background for 354 blank spots. There are 355 blank spots before the outlier deletion.

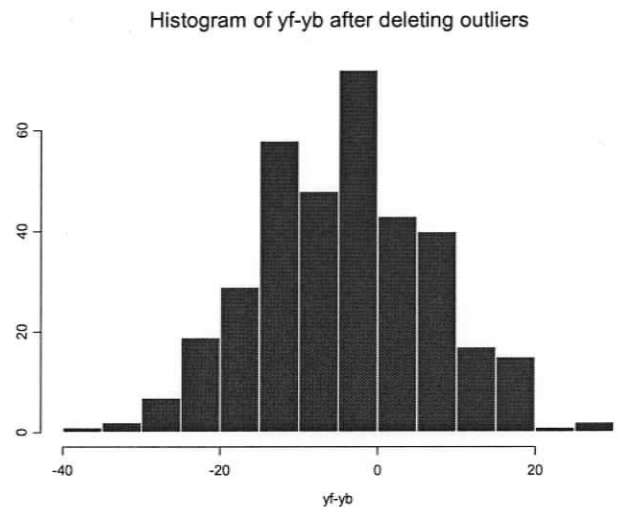

Figure 5-18 Histogram of difference between
Background and foreground median

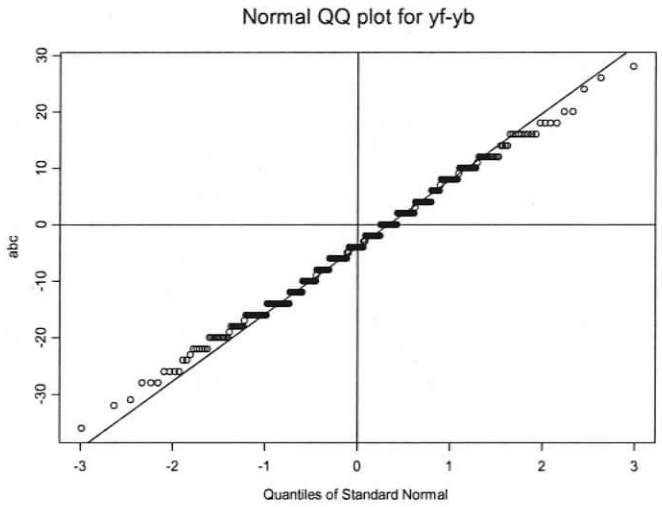

Figure 5-19 Normal QQ plot of difference
of background and foreground median

Figure 5-18 and 5-19 suggest that the difference between background and foreground median for blank spots follows normal distribution with mean zero. This satisfies the model assumptions and conditions.

- Estimate of G_0 which is the distribution of background. The maximum of the

parameter ϕ is greater than 0 and less than the maximum of y_f for the blank spot which is 336.

The number of spots is 354
The range of the ϕ is [0, 336]
$l(\hat{G}_0)$: -1597.87455055562
Num of supports m = 14

ϕ	π	$D_G(\phi)$
215.578456356552	0.00200566913291073	-0.0243964761188273
223.534222799781	0.0164545165268689	-0.00782512694664828
235.558988319824	0.115146327016704	-0.0067194127601411
245.230375187811	0.122783476985728	-0.00476164137933321
250.966126844714	0.10600382148212	-0.00702798472031696
257.752703828105	0.0368031127253619	-0.00517027702793982
262.812661085165	0.152563458593739	-0.00751119053732141
272.912397795071	0.135926222444792	-0.013568822707669
282.119356227474	0.0847628469923043	-0.0012999066408721
287.569022338745	0.0992866645920879	-0.00429341969812214
299.687382142143	0.0502259500115797	-0.00504206574675115
313.549080628796	0.052025946406514	-0.0101202442545647
318.380373292856	0.0232994624874026	-0.00799933546587106
335.839083430334	0.00288906286161876	-0.0128154890811567

From the above result, it shows that the distribution of G_0 can be described as mixture model of 14 normal distributions with respective means ϕ 's and proportion π 's.

- Histogram of $\hat{S}$

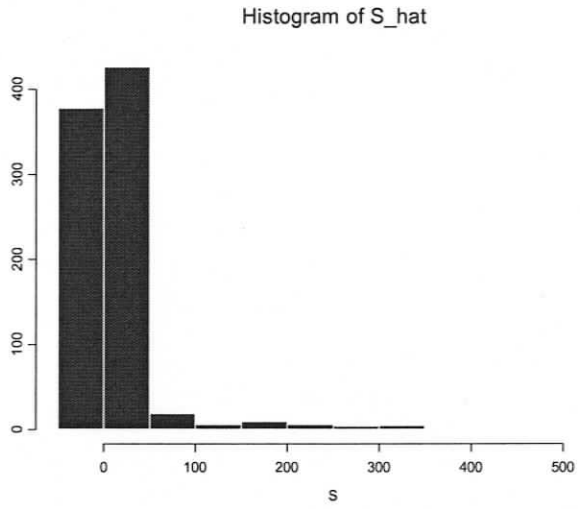

Figure 5-20 Histogram of estimate $\hat{S}$

Figure 5-20 shows the estimate of the leakage between foreground and background. Note that there are negative values. The following is the statistical summary of estimate of $\hat{S}$.

- Summary of $\hat{S}$

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-5.8349	-0.9928	0.2669	12.5243	2.7099	518.1609

- Plot of $\hat{S}$ versus y_f

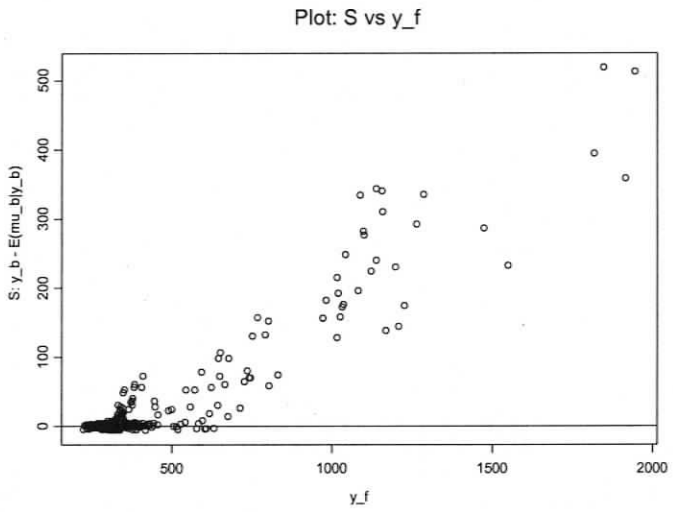

Figure 5-21 Plot of $\hat{S}$ versus y_f

Figure 5-21 shows the relationship of the leakage between spot/foreground and background and the pixel intensity of the spot/foreground. It seems in some lowly expressed genes, the spot/foreground leakage is smaller than that of the leakage from background to foreground, therefore there are some $\hat{S} < 0$. As y_f increases, the leakage of spot/foreground is larger than that of the leakage from background to foreground. $\hat{S}$ tends to increase with y_f.

- Plot of $\hat{\mu}_f$ versus $y_f - y_b$

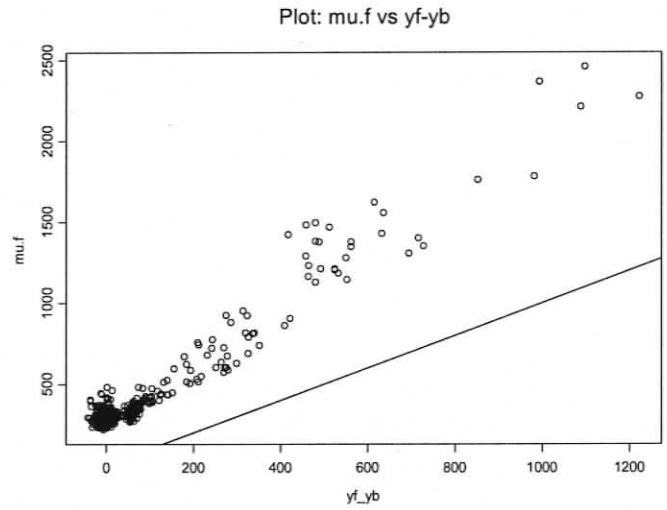

Figure 5-22 Plot of $\hat{\mu}_f$ versus $y_f - y_b$

Figure 5-22 show the relationship between $\hat{\mu}_f$ and the difference of median of the pixel intensity between foreground and background, ie. $y_f - y_b$. It shows that as $y_f - y_b$ increases, so does $\hat{\mu}_f$.

- Plot of $\hat{\mu}_f$ versus y_f

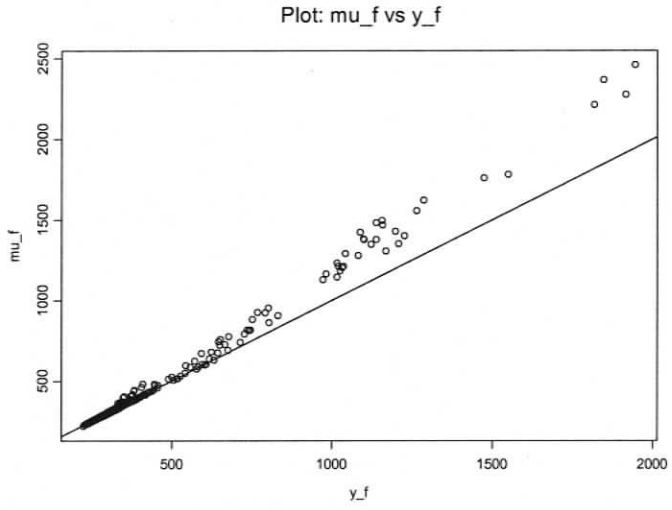

Figure 5-23 Plot of $\hat{\mu}_f$ versus y_f

Figure 5-16 shows the relationship between the mean of the corrected foreground pixel intensity and the median of foreground pixel intensity which is the uncorrected value. As it can be seen that the leakage from the foreground to background is larger than the leakage from background to foreground as y_f increases, therefore, after background correction, as y_f increases, the "true" foreground pixel intensity is larger than the uncorrected value.

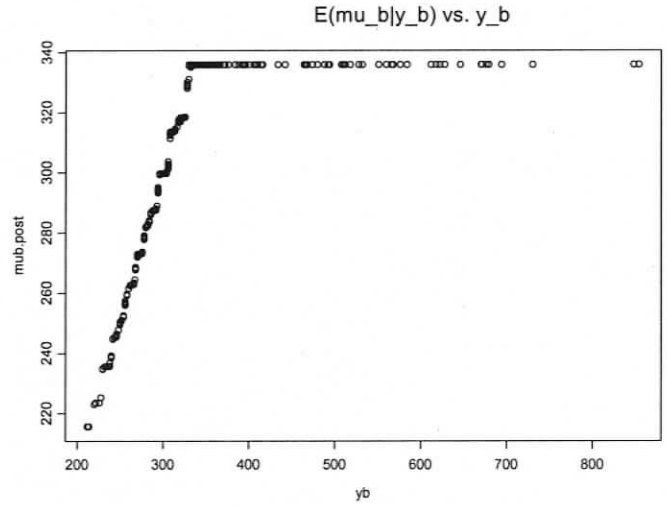

Figure 5-24 Plot of $\mathbf{E}(\hat{\mu}_b \mid y_b; \hat{G}_0, \hat{\sigma})$ versus y_b

Figure 5-17 shows the mean of the posterior of the background pixel intensities versus the background median.

Chapter 6, Discussion and Conclusions

Because of various sources of variation, the data from the lab is full of noise and random variation. The raw data should be filtered to remove some of the noise. Moreover, the minor imperfections on the array can greatly distort the background intensity and then cause unusual readouts. Background correction is a first step for data processing and it is considered a necessary step for the DNA array data analysis by many researchers. However, there is no accepted standard way to do it.

In this thesis, a new mixture model is proposed to perform background correction based on the knowledge of biochemical process underlying DNA array technology. The concept of the leakage is introduced to describe the interaction of the background and foreground. It is a mathematical way to imitate the biochemical phenomenon for the intensity of foreground and background. From the results of the model applied on the raw data set, it is consistent with what can be expected from the model as described in Chapter 3.

The model does not have the shortcomings of losing the lowly expressed gene information, or using an absolute value to floor the data. It also avoids the negative intensity problem which causes troubles with most analysis methods.

However, there are some things that still need work. Firstly, we need to examine methods for estimating the variance when using the NPMLE method to estimate the distribution of the background intensity. Secondly, when estimating the $\hat{S}$, assumptions are made on the error term which follows the normal distribution, This needs to be investigated. In addition, spatial analysis could be used to estimate the spatial variation

across the slide. In this thesis, I only consider the interaction between the background and foreground in each spot. The interaction between the strong signal spot and its neighbourhood spots should be considered in the further study. Thirdly, right now the model is consistent with the one-color macroarray data, it requires confirmation with two color DNA array data.

Bibliography

1. Bain, L.J. and Engelhardt, M. (1992) "Introduction to probability and Mathematical Statistics", Second Edition, Duxbury Thomson Leaning.
2. Bohning, D. (1986) "A Vertex-Exchange-Method in D-Optimal Design Theory", Metrika, Vol. 33, 337-347.
3. Bozinov, D. and Rahnenfuhrer, J. (2002) "Unsupervised technique for robust target separation and analysis of DNA microarray spots through adaptive pixel clustering", Bioinformatics, Vol. 18, 747-756.
4. Cooper, I.D. and Watson, G.A. (1985) "A projected Lagrangian algorithm for semi-infinite programming", Mathematical Programming, Vol. 32, No. 3, 337-356.
5. Crump, D., Werry, K., Veldhoen, N., Aggelen, G.V. and Helbing, C.C. (2002) "Exposure to the Herbicide Acetochlor Alters Thyroid Hormone-Dependent", Environmental Health Perspectives, Vol. 110, No. 12.
6. Eisen, J.A. and Hanawalt, P.C. (1999) "A phylogenomic study of DNA repair genes, proteins, and processes", Mutation Research, Vol. 435, Issue 3, 171-213.
7. Efron, B., Tibshirani, R., Goss, V. and Chu, G. (2001) "Microarrays and their use in a comparative experiment", J Amer Statist Assoc, Vol. 96, 1151-1160.
8. Fletcher, R. (1987) "Practical Method of Optimization", Second Edition, John Wiley & Sons.
9. GenePix Pro 6.0 User's Guide & Tutorial, 2004, Axon Instruments.
10. Geoffrey, J., McLachlan, K.D. and Ambroise, C. (2005) "Analyzing Microarray Gene Expression Data", John Wiley & Sons, Ltd.

11. Gill, P.E., and Murray, W. (1974) "Numerical Methods for Constrained Optimization", Academic Press.

12. Gill, P.E., Murray, W., and Wright, M.H. (1981) "Practical Optimization", Academic Press.

13. Helbing, C.C., Bailey, C.M., Ji, L., Gunderson, M., Zhang, F., Veldhoen, N., Xu, R., Lesperance, M., Holcombe, G.W., Kosian, P.A., Tietge, J., and Degitz, S.J. Differential gene expression profiles in the brain of Xenopus laevis tadpoles exposed to thyroid hormone agonists and antagonists in a metamorphosis assay (In preparation).

14. Helbing, C.C., Werry, K., Crump, D., Domanski, D., Weldhoen, N. and Bailey, C.M. (2003) "Expression Profiles of Novel Thyroid Hormone-Responsive Genes and Proteins in the Tail of Xenopus laevis Tadpoles Undergoing Precocious Metamorphosis", Molecular Endocrinology Vol. 14, Issue 7, 1395-1409.

15. ImaGene 6.0 User Manual, 2004, BioDiscovery, Inc.

16. Kooperberg, C., Fazzio, T., Delrow, J. and Tsukiyama, T. (2002) "Improved background correction fro spotted DNA microarrays", Journal of Computational Biology, Vol. 9, 55-66.

17. Legendre, P. and Legendre, L. (1996) "Numerical Ecology", Second English Edition, Elsevier.

18. Lindsay, B.G. (1998) "Mixture Model: Theory, Geometry and Applications", Institute of Mathematical Statistics, Hayward CA.

19. Lesperance, M. and Kalbfleisch, J.D. (1992) "An algorithm for computing the nonparametric MLE of a mixing distribution", Journal of the American Statistical

Association, Vol. 87,120-126.

20. Parmigiani, G., Garrett, E., Irizarry, R. and Zeger, S. (2003) "The Analysis of gene expression data: an overview of methods and software", edited by G. Parmigiani, E. Garrett, R. Irizarry, S. Zeger, Springer-Verlag, New York.
21. Petrov, A., and Shams, S. (2004) "Microarray Image Processing and Quality Control", Journal of VLSI Signal Process, Vol. 38, 211-226.
22. Pinheiro, J.C., Bates, D.M. (2000) "Mixed-Effects Models in S and S-Plus", Springer.
23. Schena, M. (2003) "Microarray Analysis", John Wiley & Sons, Inc.
24. Schermer, M.J. (1999) "Confocal scanning microscopy in microarray detection, In DNA microarrays – a practical approach Volume 1", Edited by: Schena M. Oxford: Oxford University Press.
25. Skirrow, R.C., Veldhoen, N., Ji, L., Bailey, C.M. and Helbing, C.C. "Expression profiles of thyroid hormone-induced genes in the tail of Rana catesbeiana tadpoles" (In preparation).
26. Speed, T.P. (2003) "Statistical Analysis of Gene Expression Microarray Data", Chapman & Hall/CRC.
27. Trattner, S., Greenspan, H., Tepper, G., and Abbound, S. (2004) "Automatic Identification of Bacterial Types using Statistical Imaging Methods".
28. Venables, W.N. and Ripley, B.D. (1998) "Modern Applied Statistics with S-Plus", Second Edition, Springer.
29. Veldhoen, N., Crump, D., Werry, K., and Helbing, C.C. (2002) "Distinctive Gene Profiles Occur at Key Points During Natural Metamorphosis in the Xenopus laevis Tadpole Tail", Developmental Dynamics Vol. 224, 457-468.

30. Veldhoen, N. and Helbing, C.C. (2001) "Detection of Environmental endocrine-disruptor effects on gene expression in live *Rana Catesbeliana* tadpoles using a tail fin", Environmental Toxicology and Chemistry, Vol. 20, No. 12, 2704-2708.

31. Wit, E., and McClure, J. (2005). Statistics for Microarrays: Design, Analysis and Inference, John Wiley & Sons, Ltd.

32. Wu, C.F. (1978) "Some Algorithmic Aspects of the Theory of Optimal Designs", The Annals of Statistics, Vol. 6, 1399-1412.

33. Yang, Y.H., Buckley, M.J., Dudoit, S., and Speed, T.P. (2002) "Comparison of methods for image analysis on cDNA microarray data", Journal of Computational and Graphical Statistics, Vol. 11, 1-29.

34. Zhang, F., Degitz, S.J., Holcombe, G.W., Kosian, P.A., Tietge, J., Veldohoen, N. and Helbing, C.C. (2006) "Evaluation of gene expression endpoints in the context of a Xenopus laevis metamorphosis-based bioassay to detect thyroid hormone disruptors", Aquat Toxicol. Jan. Vol. 76, Issue 1, 24-36.

Appendix

1. Matlib functions used to analyze the image

[cim, map]=imread('path\image-file.tif','tif')

- Function to read image file, if the image is color, the result is one matrix, cim, which is an $m \times n$ matrix where m is the number of row pixels and n the number of column pixels and its value is a color index which can be found in matrix, map, a 256×3 matrix, the 3 columns stand for red, blue and yellow. For example:

index	[R	G	B]	(where R, G, B $\in$ [0,1])
1	[0	0	0]	
2	[0.1	0	1]	
		:		
		:		
256	[0	1	0.5]	

So if cim(1,1)=1, that means the spot in row 1 and column 1 has the color as R=0, G=0 and B=0 which is a white spot.

c1 = ind2gray(cim,map)

- This function can change the color index to a real number, the m1 can be modified by user from the map obtained in above function

c2 = im2uint16(c1)

- This function changes the outcome of the above function to 16- bit greyscale image. 16 bit number value can be 0 – 65535. The reason to use this is to change the index color to a greyscale image to measure the intensities.

c3 = imshow(subc,map)

- This function shows the image of subc on the screen. It can be the entire image or any part of the matrix.

imresize(c3, method)

- If the above function gives a small image on the screen, one can use this function to enlarge it. If method equals 10, that means the image is enlarged to 10 times its original one.

Imwrite(image-matrix, map, filename, format)

- This function can write the imagefile which is image-matrix here with the color map to file with name filename in image format specified by format.

2. Introduction to create the DLL of the interface

2.1 Construct the minGW environment

To run the software given here, g77 and minGW (Minimalist GNU for Windows) are required. The software minGW provides a collection of freely available and freely distributable Windows specific header files and import libraries combined with GNU toolsets that allow one to produce native Windows programs that do not rely on any 3rd-party C runtime dynamic link libraries (DLLs). The download place is http://www.mingw.org/download.shtml and its official website is http://www.mingw.org/.

g77 under minGW and its environment are sufficient for our software development.

a) Download minGW from http://www.mingw.org/download.shtml

b) Install minGW

c) Modify the search path to let the g77 compiler can find the bin and lib. This can be done either by DOS command line or by modify the Environment Variables in "My Computer" -> Properties ->Advanced->"Enviroment Variables". eg.

```
SET OLDPATH=%PATH%
PATH=C:\...\MinGW\bin;C:\...\R\rw1090\bin;C:\...\R\rw1090\src\include;%
PATH%
SET LIBRARY_PATH=C:\...\MinGW\lib;C:\...\R\rw1090\lib
```

2.4 File list for the Fortran dynamic library

	Name	Type	Function
1	sp-dmngb.f	Fortran source	Library of subroutines to compute nonlinear minimization
2	sp-dnlm3b.f r-dnlm3b.f	Fortran source	Application layer creating function to be transferred to sp-dmngb.f to

			do minimization
3	param.dat	Includ function	Parameters used in file 2
4	sp-mkdll.bat	DOS batch file	Create DLL for S-Plus to call Fortran interface
5	r-mkdll.bat	DOS batch file	Create DLL for R to call Fortran interface

2.2.1 sp-dmngb.f

The program sp-dmngb.f is a modification of dmngb.f. To obtain the routine dmngb.f and all the PORT library routines it calls, send an E-mail message with subject, "send dmngb from port" to netlib@netlib.bell-labs.com. Then you will get dmngb.f and all its dependencies. These file are well tested and free to use.

This file minimizes general simply bounded objective function using analytic gradient and Hessian approximate from Secant update.

The file sp-dmngb.f is a revised version of the original one such that its I/O parts are commented out because the I/O operation is not allowed when creating the interface between S-Plus or R with Fortran subroutine.

2.2.2 sp-dnlm3b.f/r-dnlm3b.f

This file includes several subroutines to create application function and interface with sp-dmngb.f. According to the mixture model character, the linearly constrained condition can be changed to the unconstrained condition using the proper penalty function before using sp-dmngb.f. The file is listed in Appendix 4. r-dnlm3b.f is the same Fortran file but it is used to compile the dynamic link library for R.

2.2.3 param.dat

This file includes some common constants or parameters used in the above two Fortran files. If the user wants to change some parameters, s/he only need to

change the relative parameters in this file and re-compile the file to form the DLL file. There is no need to find the relative parameter in source code to make such changes.

2.3 Batch files

2.3.1 sp-mkdll.bat

This DOS batch file which contains commands to create the dynamic library sp-dnlm3b.dll which contains the Fortran interface for S-Plus. Use this batch file under DOS command window to generate the sp-dnlm3b.dll and sp-dnlm3b.def. Sqpe.dll and libsqpe.a are copied from S-Plus library({S-Plus install home directory}\lib and {S-Plus install home directory}\lib\mingw). The command list is as follows:

```
dlltool --dllname Sqpe.dll --def mingsqpe.def --output-lib libsqpe.a
   g77 -c sp-dmngb.f sp-dnlm3b.f
dlltool --export-all-symbols --output-def sp-dnlm3b.def  sp-dmngb.o sp-dnlm3b.o
dllwrap –o sp-dnlm3b.dll -- def sp-dnlm3b.def sp-dmngb.o sp-dnlm3b.o -L. -lsqpe 2>error-dll.txt
del *.o
```

2.3.2 r-mkdll.bat

This DOS batch file contains the commands to create the dynamic library r-dnlm3b.dll which contains the Fortran interface for R. The command list is as follows:

```
g77 –shared  -o  r-dnlm3b.dll  sp-dmngb.f  r-dnlm3b.f
```

2.4 Installation of DLL file

For S-Plus, copy the two files **sp-dnlm3b.dll, sp-dnlm3b.def** to the s-plus library directory {S-plus home directory}\lib\mingw\

For R, copy the file **r-dnlm3b.dll** to {R-home directory}\lib

3. List of S-Plus script file

There are two S-Plus files titled bgd_correction.ssc and backsub.ssc, the latter one provides all the functions needed in the first one. There are roughly 3 parts of bgd_correction.ssc: setup and data cleaning, estimation of G_0 and background correction. If you do not want to change the algorithm, you do not need to edit backsub.ssc. The places that the user must change when they begin to use the script are shown in bold characters. The following is the S-Plus script bgd_correction.ssc.

```
#************* Step 1 - set up data and environment ***************
# Start the program here !!

# Obtain current working directory, user designate the work path where
# the result file will be stored. User also can this script
# and backsub.ssc in this home directory
# The following function is to obtain the home directory
#
getpwd <- function() {
#  return("{Your working directory }")
#  Please change next line to return your work directory
        return("D:\\Math-Soft\\Project\\S-Plus\\s-fortran")
}

# Load the dynamic link library which includes the Fortran subroutine.
# User can put this *.dll anywhere, generally it will be put
# in the lib of splus which will be like
# {S-Plus Home directory}/lib/mingw/
loadlib<-function() {
#  dll.load("{S-Plus Home directory}\\lib\\mingw\\sp-dnlm3b.dll","dnlm3b_")
#  Please make necessary change to the next line with the format above.
   dll.load("C:\\My-App\\Insightful\\splus62se\\lib\\mingw\\sp-dnlm3b.dll","dnlm3b_")
}

# Obtain the workpath and the data file name for y.dat and size.dat
 workpath <- getpwd()
 ydata <- paste(workpath, "\\y.dat",sep="")
 sizedata <- paste(workpath, "\\size.dat",sep="")

#
# User should designate the storate place of backsub.ssc here.
# Either you copy the file to a any place, saying "c:/Temp/s-plus/backsub.ssc"
#   and designate backsub.file<-" c:/Temp/s-plus/backsub.ssc "
# Or just copy it to your work directory and use the next line
backsub.file<-paste( workpath, "\\backsub.ssc", sep="")
source(file=backsub.file,local=F)
```

```
#
#*********************************************************
# Create output and plots for work.dataset array 2 in paper

#
# Prepare dataset
# Either you can designate the file directory as the next line
#   datafile_"c:/temp/tmp/Caren-data/epa-b/t31h24a16na.txt"
# Or store your data in the data directory and use the next two lines
dataset.path<-"c:/temp/tmp/Caren-data/epa-b"
datafile_paste(dataset.path, "/t31h24a16na.txt",sep="");

# Designate data name and work title to present in graphs
data.name_"t31h24a16na.lst"
work.title<-"epa-b: t31h24a16na"

#########################################################
#  Users needs only to change the directory and dll path
#  above if they do not want to change the algotithms
#########################################################
```

The user can modify some set of I/O of S-Plus after seeing the above lines.

4. Fortran code

The Fortran code can be found in netlib.bell-labs.com. The modification of sp-dmngb.f is to comment out all the IOs in the original file dmngb.f.

5. S-Plus/R code

5.1 S-Plus script files

5.1.1 bgd_correction.ssc

```
#************* Step 1 - set up data and environment ***************************
# Start the program here !!

# Obtain current working directory, user designate the work path where
# the result file will be stored. User also can this script
# and backsub.ssc in this home directory
# The following function is to obtain the home directory

getpwd <- function() {
#  return("{Your working directory }")
```

```
#  Please change next line to return your work directory
    return("D:\\Math-Soft\\Project\\S-Plus\\s-fortran")
}

# Load the dynamic link library which includes the Fortran subroutine.
# User can put this *.dll anywhere, generally it will be put
# in the lib of splus which will be like
# {S-Plus Home directory}/lib/mingw/
loadlib<-function() {
#  dll.load("{S-Plus Home directory}\\lib\\mingw\\sp-dnlm3b.dll","dnlm3b_")
#  Please make necessary change to the next line with the format above.
   dll.load("C:\\My-App\\Insightful\\splus62se\\lib\\mingw\\sp-dnlm3b.dll","dnlm3b_")
}

# Obtain the workpath and the data file name for y.dat and size.dat
 workpath <- getpwd()
 ydata <- paste(workpath, "\\y.dat",sep="")
 sizedata <- paste(workpath, "\\size.dat",sep="")

# User should designate the storate place of backsub.ssc here.
# Either you copy the file to a any place, saying "c:/Temp/s-plus/backsub.ssc"
#   and designate backsub.file<-" c:/Temp/s-plus/backsub.ssc "
# Or just copy it to your work directory and use the next line
backsub.file<-paste( workpath, "\\backsub.ssc", sep="")
source(file=backsub.file,local=F)

#*****************************************************************
# Create output and plots for work.dataset array 2 in paper

#
# Prepare dataset
# Either you can designate the file directory as the next line
#    datafile_"c:/temp/tmp/Caren-data/epa-b/t31h24a16na.txt"
# Or store your data in the data directory and use the next two lines
dataset.path<-"c:/temp/tmp/Caren-data/epa-b/from-raw-data"
datafile_paste(dataset.path, "/t3-0.31/t3-0.31h24a14na.txt",sep="");

# Designate data name and work title to present in graphs
data.name_"t31h24a14na.lst"
work.title<-"epa-b: t31h24a14na"

############################################################
# Users needs only to change the directory and dll path
# above if they do not want to change the algotithms
############################################################
# Get the raw data and clean the data
org.data_data.prep(datafile)

dataqf_org.data[["data.qf"]]
blk_dataqf[dataqf$labels == "Blank",]
nblk_dataqf[dataqf$labels != "Blank",]

# If using background to estimate the distribution of background,
# blkindex should be set to "T" to let function call the backgrdsub()
blkindex <- T
work.dataset <- blk
```

```
### delete the spot near saturated spot.
cc_cls.blk(nblk,blk)
work.dataset_work.dataset[cc,]

# Set up the work database
work.dataset<-work.dataset[order(work.dataset$labels),]

#################### Remove outliers in blank spots
# Delete outliers in yf
# If the yf does not have significan outliers, no need to run the following 5 lines
abc_work.dataset$yf
cbc_rm.outliers(abc)
ma_mean(abc[cbc])
work.dataset_work.dataset[cbc,]
length(work.dataset$yf)

# Delete outlier in yb
# If the yb does not have significan outliers, no need to run the following 5 lines
abc_work.dataset$yb
cbc_rm.outliers(abc)
ma_mean(abc[cbc])
work.dataset_work.dataset[cbc,]
length(work.dataset$yf)

######## may not need to delete outliers for yf & yb
######## separately and do the following only
# delete outlier in y_f-y_b
abc_work.dataset$yf-work.dataset$yb
hist(abc,main="Histogram of y_f-y_b before deleting outliers")
qqnorm(abc)
qqline(abc)
cbc_rm.outliers(abc)
abc_abc[cbc]
hist(abc,xlab="y_f-y_b",main="Histogram of y_f-y_b after deleting outliers")
ma_mean(abc)
qqnorm(abc,main="Normal QQ plot for y_f-y_b")
qqline(abc)
work.dataset_work.dataset[cbc,]
length(work.dataset$yf)

abline(h=0,v=0)
#################### end of Remove outliers

# Designate the bad spot so that to get rid of them in the late analysis
bad.spot<-c(3,5)
#******** End of Step 1 - run to here ******************


#******** Step 2 - estimate G0 by blank spots ***********
X<-max( work.dataset$yf )

#Checking distribution of yf for blank spots
summary(work.dataset$yf)
hist(work.dataset$yf)
```

```
# Get the upper boundary of the estimate range
xmaxtitle<-paste("X =", X)

if( blkindex ) {
  work.data.max<-backgrdsub(work.dataset,flag.bad=bad.spot,max.dif=X,
      mytitle=work.title, yfile=ydata, sizefile=sizedata)
} else {
  work.data.max<-backsub(work.dataset,flag.bad=bad.spot,max.dif=X,
          mytitle=work.title, yfile=ydata, sizefile=sizedata)
}

# run fortran programme
tablefile <- paste(workpath, "\\npar.lst",sep="")
fortrancall(0,X, ydata,sizedata,tablefile)

#filename <- paste(workpath, "\\dataset\\", data.name, sep="")
dos.copy(data.name)

#********** End of Step 2 *****************

#********** Start of Step 3: Background correction *******
# read in y_b for non-blank spots
# rawdata has most columns from Imagene dataset

raw_org.data[["rawdata"]]
 raw.nonblank_raw[raw$labels!="Blank",]
tmpraw_data.oper(raw.nonblank,flag.bad=bad.spot)

# The following two lines set up the working dataset
work.dataset_tmpraw
a_work.dataset
X<-max( work.dataset$yb )
xmaxtitle<-paste("X =", X)
work.data.max<-backsub(work.dataset,flag.bad=bad.spot,
   max.dif=X,mytitle=work.title, yfile=ydata, sizefile=sizedata)
filename_tablefile

# Compute the posterior
sm.max<-postest(work.data.max$ydif, work.data.max$c.sigma, work.data.max$c.sizes,
      work.data.max$data, tablefile=filename)

#Problem - why are there NA's in posterior?  Because outliers y_b 12,582, 12,226
b_sm.max$post

# The following lines are to get rid of the NA spots
cna_b[is.na(b)]
cc_is.na(b)
ind.na_(1:length(b))[cc]
a$yb[ind.na]
a$yf[ind.na]
length(a$yf)
yb_a$yb
yf_a$yf

if( length(ind.na) > 0 ) {
   yb_yb[-ind.na]
```

```
    yf_yf[-ind.na]
    mub.post_b[-ind.na]
    atmp_a[-ind.na,]
  } else {
    mub.post_b
    atmp_a
  }

  # Get the interaction of background and foreground
  b.diff_(yb-mub.post)
  hist(b.diff, xlab="S", main="Histogram of S_hat")
  summary(b.diff)

  plot(yb, mub.post, main="yb-s-hat vs. y_b")
  qqnorm(b.diff)
  qqline(b.diff)

 # Plot of S_hat vs. yf
  c1_yf > 0
  cltit_"S_hat(=y_b - post) vs y_f"

  ind.reg_(1:length(yf))[c1]
  yf3_yf[c1]
  dif3_b.diff[c1]
  plot(yf3,dif3,main="Plot: S vs y_f ",
     xlab="y_f", ylab="S: y_b - E(mu_b|y_b) ")
  abline(0,0)

 # Plot of S_hat vs. yf in log scale
  cltit_"Plot: S' vs ln(y_f)"
  yf3_log(yf[c1])
  dif3_(log(yb)-log(mub.post))[c1]

  plot(yf3,dif3,main="Plot: S' vs ln(y_f) ",
     xlab="ln(y_f)", ylab="S'=ln(yb)-ln(E(mu_b|y_b))")
  abline(0,0)

  plot(yb, mub.post, main="E(mu_b|y_b) vs. y_b" )

  # Target: estimate of "true" foreground
  mu.f_yf+b.diff

  # plot for the mu.f vs. yf
  plot(yf, mu.f, main="Plot: mu_f vs y_f",ylab="mu_f",xlab="y_f")
  abline(0,1)

  # Write back the estimate "true" value to dataset with column name "mu.f"
  tmpbgcor_cbind(atmp, mu.f)
  nn_length(names(tmpbgcor))
  names(tmpbgcor)[nn]_"mu.f"

  nvec_rep(NA,length(raw$yf))
  bgcor_cbind(raw,nvec)
  nn_length(names(bgcor))
  names(bgcor)[nn]_"mu.f"
```

```
bgcor[tmpbgcor$no,]$mu.f_tmpbgcor$mu.f

# Cleanup
# The following two lines are the output for further study.
out.bgcor<-bgcor
out.muf<-mu.f

# Clean up
alist<-objects()
rm(list=alist)
rm(alist)

#************** End of Step 3 ************************
##########################################################
```

5.1.2 **backsub.ssc**

```
backsub<-function(dataset, flag.bad = c(3,5), max.dif = 1000, min.dif = -5000,
      var.med = "observed", possible.betas = c(0.01, 10),
      yfile = ydata, sizefile = sizedata,mytitle = "")
{
 ### M. Lesperance, University of Victoria, 2004
 ### Modified by Edward Chang, November 2004

 ### dataset is a data frame containing 10 columns:
 ###   labels:  gene names of spots **ensure that these are accurate!!
 ###   xf and xb, and yf and yb: means and medians of foreground and
 ###     background intensity for each spot
 ###   nf and nb: number of pixels in the foreground and background of each spot
 ###   sf and sb: standard deviation of foreground and background intensity of each spot
 ###   flag: an indicator as to whether each spot is acceptable
 ### flag.value.bad:  vector containing values of flag associated with bad spots
 ### max.dif: maximum difference between foreground and background median
 ###    intensity for which further analysis is performed - otherwise raw estimate is retained
 ### min.dif: smallest acceptable difference for which further analysis is performed
 ### var.med: if 'observed'  variance of median under distributional assumptions,
 ###   is computed using sample median; otherwise expected under model
 ### possible.betas:  values of beta to consider for weibull distribution
 ### yfile : file name for yt data for fortran program
 ### sizefile: file name for standard deviation estimates and custer sizes for each gene cluster
 ### returns from original data
    ## data = original data, sorted by labels, attached c.num (cluster num), ydif, b.min, b.max,
     b.rem
 ## pF = F tests for original data clusters by labels
 ## returns for analyzed data - ydif, c.sigma (cluster sigmas), c.sizes (cluster sizes), c.num
 # order dataset by gnames ** ensure labels are characters NOT factors
 dataset$labels <- as.character(dataset$labels)
 dataset <- dataset[order(dataset$labels), ]
 ##
 # F-test for equality of foreground means by clusters defined by gnames
 #   provides a check on the integrity of gene clusters
 ##
 c.sizes <- psizes(dataset$labels)

 pF <- pftest(dataset$xf, dataset$sf, dataset$nf, c.sizes)
 dataset$c.num <- rep(1:length(c.sizes), c.sizes)
```

```
  ##
  # note points with yf-yb<min.dif or yf-yb>max.dif or with flag=flag.bad and remove from analysis
  ##

  dataset$ydif <- dataset$yb
  dataset$b.min <- dataset$ydif < min.dif
  dataset$b.max <- dataset$ydif > max.dif
  dataset$b.rem <- dataset$b.min | dataset$b.max | (is.element(dataset$flag,flag.bad))

  dataf <- dataset[!dataset$b.rem, ]
  ##
  # compute cluster sample statistics
  ##
  c.sizes <- psizes(dataf$labels)
  xf <- psum(dataf$xf, c.sizes)/c.sizes
  xb <- psum(dataf$xb, c.sizes)/c.sizes
  yf <- psum(dataf$yf, c.sizes)/c.sizes
  yb <- psum(dataf$yb, c.sizes)/c.sizes
  sf <- psum(dataf$sf, c.sizes)/c.sizes
  sb <- psum(dataf$sb, c.sizes)/c.sizes
  nf <- psum(dataf$nf, c.sizes)/c.sizes
  nb <- psum(dataf$nb, c.sizes)/c.sizes
  ##
  # model.selector estimates standard deviations for each foreground and
  # background median used to estimate the standard deviation for each dif
  ##
  cat("foreground:", length(xf), mean(nf), "  ")
  model.f <- modelselec(xf, yf, sf, nf, var.med, possible.betas, paste(mytitle, " Foreground"))
  cat("background:", length(xb), mean(nb), "  ")
  model.b <- modelselec(xb, yb, sb, nb, var.med, possible.betas, paste(mytitle, " Background"))

#######  20050512
  c.sigma <- sqrt(model.b$sigma^2)
###         c.sigma <- sqrt(model.f$sigma^2 + model.b$sigma^2)
#######  20050512

  ##
  # write dif, sigma and cluster sizes for clusters used in non-parametric MLE calc
  ##
  #write.table(dataf$ydif, file = yfile) ## does not work with unix system
  for(i in 1:length(dataf$ydif)) {
          cat(dataf$ydif[i], "\n", file = yfile, append = i > 1)
  }
  #write.table(cbind(c.sigma, c.sizes), dimnames.write = F, sep = " \t", file = sizefile)
  for(i in 1:length(c.sizes)) {
          cat(c.sigma[i], "\t", c.sizes[i], "\n", file = sizefile, append = i > 1)
  }
  return(list(data = dataset, ydif = dataf$ydif, c.sigma = c.sigma, c.sizes = c.sizes, c.num =
          dataf$c.num, pF = pF))
}

backgrdsub<-
function(dataset, flag.bad = c(3,5), max.dif = 1000, min.dif = -5000,
     var.med = "observed", possible.betas = c(0.01, 10),
     yfile = ydata, sizefile = sizedata,mytitle = "")
{
```

```
### Modified by Edward Chang, November 2004
### This is a variation of backsub
### Using yf for blank spot background estimation to
### reduce the variance

### dataset is a data frame containing 10 columns:
###   labels:  gene names of spots **ensure that these are accurate!!
###   xf and xb, and yf and yb: means and medians of foreground and
###     background intensity for each spot
###   nf and nb: number of pixels in the foreground and background of each spot
###   sf and sb: standard deviation of foreground and background intensity of each spot
###   flag: an indicator as to whether each spot is acceptable
### flag.value.bad:  vector containing values of flag associated with bad spots
### max.dif: maximum difference between foreground and background median
###    intensity for which further analysis is performed - otherwise raw estimate is retained
### min.dif: smallest acceptable difference for which further analysis is performed
### var.med: if 'observed'  variance of median under distributional assumptions,
###   is computed using sample median; otherwise expected under model
### possible.betas:  values of beta to consider for weibull distribution
### yfile : file name for yt data for fortran program
### sizefile: file name for standard deviation estimates and custer sizes for each gene cluster
###
### returns from original data
## data = original data, sorted by labels, attached c.num (cluster num), ydif, b.min, b.max, b.rem
## pF = F tests for original data clusters by labels
## returns for analyzed data - ydif, c.sigma (cluster sigmas), c.sizes (cluster sizes), c.num
# order dataset by gnames ** ensure labels are characters NOT factors
dataset$labels <- as.character(dataset$labels)
dataset <- dataset[order(dataset$labels),  ]
##
# F-test for equality of foreground means by clusters defined by gnames
#   provides a check on the integrity of gene clusters
##
c.sizes <- psizes(dataset$labels)

pF <- pftest(dataset$xf, dataset$sf, dataset$nf, c.sizes)
dataset$c.num <- rep(1:length(c.sizes), c.sizes)
##
# note points with yf-yb<min.dif or yf-yb>max.dif or with flag=flag.bad and remove from analysis
##

dataset$ydif <- dataset$yf

dataset$b.min <- dataset$ydif < min.dif
dataset$b.max <- dataset$ydif > max.dif

dataset$b.rem <- dataset$b.min | dataset$b.max | (is.element(dataset$flag,flag.bad))

dataf <- dataset[!dataset$b.rem,  ]
##
# compute cluster sample statistics
##
c.sizes <- psizes(dataf$labels)
xf <- psum(dataf$xf, c.sizes)/c.sizes
xb <- psum(dataf$xb, c.sizes)/c.sizes
yf <- psum(dataf$yf, c.sizes)/c.sizes
```

```
  yb <- psum(dataf$yb, c.sizes)/c.sizes
  sf <- psum(dataf$sf, c.sizes)/c.sizes
  sb <- psum(dataf$sb, c.sizes)/c.sizes
  nf <- psum(dataf$nf, c.sizes)/c.sizes
  nb <- psum(dataf$nb, c.sizes)/c.sizes
  ##
  # model.selector estimates standard deviations for each foreground and
  # background median used to estimate the standard deviation for each dif
  ##
  cat("foreground:", length(xf), mean(nf), "  ")
  model.f <- modelselec(xf, yf, sf, nf, var.med, possible.betas, paste(mytitle, " Foreground"))
  cat("background:", length(xb), mean(nb), "  ")
  model.b <- modelselec(xb, yb, sb, nb, var.med, possible.betas, paste(mytitle, " Background"))

   ktmp_nf/(nf+nb)
  c.sigma <- sqrt(model.f$sigma^2)

  ##
  # write dif, sigma and cluster sizes for clusters used in non-parametric MLE calc
  ##
  for(i in 1:length(dataf$ydif)) {
          cat(dataf$ydif[i], "\n", file = yfile, append = i > 1)
  }
  #write.table(cbind(c.sigma, c.sizes), dimnames.write = F, sep = " \t", file = sizefile)
  for(i in 1:length(c.sizes)) {
          cat(c.sigma[i], "\t", c.sizes[i], "\n", file = sizefile, append = i > 1)
  }
  return(list(data = dataset, ydif = dataf$ydif, c.sigma = c.sigma, c.sizes = c.sizes, c.num =
          dataf$c.num, pF = pF))
}


postest<-
function(ydif, c.sigma, c.sizes, dataset, tablefile = "D:/Math-Soft/Project/S-Plus/s-fortran/npar.lst")
{
  ### Mary Lesperance, August 2004
  ### Modified by Edward Chang, November 2004

  ### Calculates posterior means for each ydif given cluster sigmas and Ghat
  ###  assumes ydif | mu ~ normal(mu,sigma);  mu ~ Ghat
  ### ydif = differences yf-yb which satisfy inclusion criteria
  ### c.sigma = vector of cluster standard deviations of medians
  ### c.sizes = vector of cluster sizes for the data
  ### dataset = original dataset - updated with posterior estimates
  ### tablefile =  filename where the fortran output for Ghat; need thetas, weights
  ### read details of prior distribution (non-parametric maximum likelihood distribution) from file
  ## returns post = posterior means for spots that satisfy inclusion criteria
  ## data = dataset, with column ydif - spots that satisfy inclusion criteria are updated with
  ##         posterior means
  Ghat <- read.table(tablefile, header = TRUE, skip = 2)
#mu <- Ghat$thetas - ma/2
  mu <- Ghat$thetas
  probs <- Ghat$weights
  n <- length(ydif)
  k <- length(mu)
  vmu <- rep(mu, n)
```

```
 vprobs <- rep(probs, n)
 sigma <- rep(c.sigma, c.sizes)
 vsigma <- rep(sigma, rep(k, n))
 vydif <- rep(ydif, rep(k, n))
 vps <- dnorm(vydif, vmu, vsigma)
 num <- apply(matrix(vmu * vps * vprobs, nrow = n, ncol = k, byrow = T), 1, sum)
 denom <- apply(matrix(vps * vprobs, nrow = n, ncol = k, byrow = T), 1, sum)
 post <- num/denom

### 20050512
 dataset$ydif[!dataset$b.rem] <- post
### 20050512

 return(list(post = post, data = dataset))
}

modelselec<-
function(x, y, s, n, var.med = "observed", possible.betas = c(0.01, 10), mytitle = "")
{
 ### Karissa Johnston, Mary Lesperance - August 2004
 ### Modified by Edward Chang, November 2004

 ### This function takes vectors of means, standard deviations, and
 ###    sample sizes, and uses them to produce moment estimators
 ###    of parameters from four parametric distributions. The parametric medians
 ###    are moment estimated and compared to the observed
 ###    medians.  The distribution which produces the best agreement between
 ###    estimated and observed medians using an R-squared-type value
 ###    is chosen as the best model. The distribution chosen and
 ###    the estimated variance of the median given this distribution are returned.
 ###  If the moment estimator of the variance is negative, this point is
 ###    omitted from choosing a distribution, but the standard
 ###    deviation is then set to the mean of all the other standard deviations.
 ###
 ### x = vector of means
 ### y = vector of medians
 ### s = vector of standard deviations
 ### n = vector of sample sizes
 ### if var.meds= "observed", variance is calculated using observed medians,
 ###    otherwise expected medians.
 var.y <- var(y)

 ##
 # Gamma
 ##
 distn <- "Gamma"
 temp <- gam.model(x, y, s, n, var.med)
 med.gam <- temp$med
 med <- med.gam
 sigma <- temp$sigma
 R <- 1 - mean((y - med.gam)^2, na.rm = T)/var.y
 plot(y, med.gam)
 abline(0, 1)
 title(paste(mytitle, " Gamma"))

 ##
```

```
# Weibull
##
temp <- weibull.model(x, y, s, n, var.med, possible.betas)
temp <- new.weibull.model(x, y, s, n, var.med, possible.betas)
med.weib <- temp$med
R.weibull <- 1 - mean((y - med.weib)^2, na.rm = T)/var.y
plot(y, med.weib)
abline(0, 1)
title(paste(mytitle, " Weibull"))
if(R.weibull > R) {
        distn <- "weibull"
        med <- med.weib
        R <- R.weibull
        sigma <- temp$sigma
}

##
# Log-normal
##
temp <- lnorm.model(x, y, s, n, var.med)
med.lnorm <- temp$med
R.lnorm <- 1 - mean((y - med.lnorm)^2, na.rm = T)/var.y
plot(y, med.lnorm)
abline(0, 1)
title(paste(mytitle, " Log-normal"))
if(R.lnorm > R) {
        distn <- "lognormal"
        med <- med.lnorm
        R <- R.lnorm
        sigma <- temp$sigma
}

##
# Inverse Gaussian
##
#temp <- invgauss.model(x, y, x, n, var.med)
#med.invg <- temp$med
#R.invg <- 1 - mean((y - med.invg)^2, na.rm = T)/var.y
#plot(y, med.invg)
#abline(0, 1)
#title(paste(mytitle, " Inverse Gaussian"))
#if(R.invg > R) {
#       distn <- "inverse gaussian"
#       med <- med.invg
#       R <- R.invg
#       sigma <- temp$sigma
#}
if(sum(is.na(sigma)) > 0) {
        #               index <- c(1:length(sigma))
        #               indicator <- index * is.na(sigma)
        #               sigma[indicator] <- mean(sigma, na.rm = T)
        indicator <- is.na(sigma)
        sigma[indicator] <- mean(sigma, na.rm = T)
        cat("following sigmas were NA (probably sigma squ. was estimated to be negative):
sigma index number",
                indicator[indicator > 0], "\n", "they were set to the mean of all other sigmas",
```

```
                "\n")
        change <- 1
 }
 else change <- 0
 cat("R = ", R, "\t", "distn = ", distn, "\n")
 plot(y, med, xlab = "observed medians", ylab = "moment estimates of medians")
 title(paste(mytitle, distn))
 abline(0, 1)
 if(change == 1)
        points(y[indicator], med[indicator], pch = 15)
 return(distn, R, sigma)
}

pftest<-
function(xbars, ss, ns, c.sizes)
{
 # Mary Lesperance, August 2004
 # Computes F-test for equality of group means given cluster of sizes c.sizes
 # returns 1 for clusters of size 1
 # xbars = sample means; ss = sample standard deviations; ns = n's for each sample
 # c.sizes = cluster/group sizes
 # Note:  sum(c.sizes) must equal length(xbars)=length(ss)=length(ns)
 c.N <- psum(ns, c.sizes)
 c.avg <- psum(ns * xbars, c.sizes)/c.N
 MSB <- Div(psum(ns * (xbars - rep(c.avg, c.sizes))^2, c.sizes), (c.sizes - 1))
 MSW <- Div(psum((ns - 1) * ss^2, c.sizes), (c.N - c.sizes))
 dfnum <- (c.sizes - 1) + (c.sizes == 1)
 dfden <- (c.N - c.sizes) + (c.sizes == 1)
 c.Ftest <- 1 - pf(Div(MSB, MSW), dfnum, dfden)
 return(c.Ftest)
}

psum<-
function(vec, sizes)
{
 # Mary Lesperance, August 2004
 ## partition sum of vec having clusters of sizes
 res <- tapply(vec, list(rep(1:length(sizes), sizes)), sum)
 return(res)
}

psizes<-
function(vec)
{
 # Mary Lesperance, August 2004
 # create vector of cluster sizes given vector of cluster numbers
 # Modified by Edward Chang, November 2004

 inblk <- vec[ vec != "Blank" ]
 iblk  <- vec[ vec == "Blank" ]
 nblk  <- length( iblk )

 if( nblk != 0 )
 {
   iblk <- paste("Blank",1:nblk)
```

```
    vec1 <- c( as.character(inblk), as.character(iblk) )
    vec1 <- vec1[order(vec1)]
    vec  <- vec1
  }

 res <- tapply(vec, list(vec), length)
 return(res)
}

Div<-
function(a, b)
{
 # Mary Lesperance, August 2004
 #divide where division by zero returns zero
 (a/(b + (b == 0))) * (b != 0)
}

gam.model<-
function(x, y, s, n, var.med)
{
 # Karissa Johnston, August 2003
 ### This function takes vectors of means,standard deviations,and sample sizes.
 ### It assumes each sample comes from the gamma distribution, gets moment ests of parameters
for each sample,
 ### and uses these to estimate the median and its variance for each sample
 ### x is a vector of means
 ### y is the vector of observed medians
 ### s is a vector of standard deviations
 ### n is a vector of sample sizes
 z <- (n - 1)/n
 theta <- (z * s^2)/x
 kappa <- x/theta
 med <- c()
 sigma <- c()
 for(i in 1:length(x)) {
          med[i] <- qgamma(0.5, shape = kappa[i], rate = 1/theta[i])
          if(var.med == "observed")
                  sigma[i] <- 0.5/(sqrt(n[i]) * dgamma(y[i], shape = kappa[i], rate = 1/theta[
                          i]))
          else sigma[i] <- 0.5/(sqrt(n[i]) * dgamma(med[i], shape = kappa[i], rate = 1/theta[
                          i]))
 }
 return(med, sigma)
}

weibull.model<-
function(x, y, s, n, var.med, possible.betas)
{
 # Karissa Johnston, August 2003
 ### This function takes vectors of means,standard deviations,and sample sizes.
 ### It assumes each sample comes from the weibull distribution, gets moment estimators of
parameters for each sample,
 ### and uses these to estimate the median and its variance for each sample
 ### x is a vector of means
 ### y is a vector of observed medians
 ### s is a vector of standard deviations
```

```
  ### n is a vector of sample sizes
  ### possible.betas is the range of allowable values for beta
  z <- (n - 1)/n
  f <- function(x, s, z, beta)
  {
          return(abs(gamma(1 + (2/beta))/(gamma(1 + (1/beta))^2) - (((z * s^2)/x^2) + 1)))
  }
  beta <- c()
  min.diff <- c()
  for(i in 1:length(x)) {
          temp <- optimize(f, interval = possible.betas, x = x[i], s = s[i], z = z[i])
          beta[i] <- temp$minimum
          min.diff[i] <- temp$objective
  }
  theta <- x/gamma(1 + (1/beta))
  med <- qweibull(0.5, beta, theta)
  if(var.med == "observed")
          sigma <- 0.5/(sqrt(n) * dweibull(y, beta, theta))
  else sigma <- 0.5/(sqrt(n) * dweibull(med, beta, theta))
  return(med, sigma, min.diff, beta)
}

new.weibull.model<-
function(x, y, s, n, var.med="observed")
{

  ### This function takes vectors of means,standard deviations,and sample sizes.
  ### It assumes each sample comes from the weibull distribution, gets moment estimators of
parameters for each sample,
  ### and uses these to estimate the median and its variance for each sample
  ### x is a vector of means
  ### y is a vector of observed medians
  ### s is a vector of standard deviations
  ### n is a vector of sample sizes

  z <- (n - 1)/n
  beta <- c()
  min.diff <- c()
  gridbeta<-seq(0.02,10,by=0.001)
  G1<-gamma(1+(2/gridbeta))
  G2<-gamma(1+(1/gridbeta))^2
  for(i in 1:length(x)) {
                        fvec<-abs((G1/G2)-(((z[i]*s[i]^2)/x[i]^2)+1))
                        beta[i]<-gridbeta[fvec==min(fvec)]
                        min.diff[i]<-min(fvec)

                        }
  theta <- x/gamma(1 + (1/beta))
  med <- qweibull(0.5, beta, theta)
  if(var.med == "observed")
          sigma <- 0.5/(sqrt(n) * dweibull(y, beta, theta))
  else sigma <- 0.5/(sqrt(n) * dweibull(med, beta, theta))
  return(med,sigma,min.diff,beta,theta)


}
```

```
lnorm.model<-
function(x, y, s, n, var.med)
{
 # Karissa Johnston, August 2003
 ### This function takes vectors of means,standard deviations,and sample sizes.
 ### It assumes each sample comes from the lognormal distribution, gets moment estimators of
parameters for each sample,
 ### and uses these to estimate the median and its variance for each sample
 ### x is a vector of means
 ### y is a vector of medians
 ### s is a vector of standard deviations
 ### n is a vector of sample sizes
 z <- (n - 1)/n
 sig.squ <- log(((z * s^2)/x^2) + 1)
 mu <- log(x) - sig.squ/2
 med <- exp(mu)
 if(var.med == "observed")
        sigma <- 0.5/(sqrt(n) * dlnorm(y, mu, sqrt(sig.squ)))
 else sigma <- 0.5/(sqrt(n) * dlnorm(med, mu, sqrt(sig.squ)))
 return(med, sigma)
}

invgauss.model<-
function(x, y, s, n, var.med)
{
 # Karissa Johnston, August 2003
 ### This function takes vectors of means,standard deviations,and sample sizes.
 ### It assumes each sample comes from the inverse gaussian distribution, gets moment estimators
of parameters for each sample,
 ### and uses these to estimate the median and its variance for each sample
 ### x is a vector of means
 ### y is a vector of observed medians
 ### s is a vector of standard deviations
 ### n is a vector of sample sizes
 z <- (n - 1)/n
 mu <- x
 lambda <- x^3/(z * (s^2))
 med <- c()
 sigma <- c()
 med <- qinvgauss(rep(0.5, length(x)), mu, lambda)
 if(var.med == "observed")
        sigma <- 0.5/(sqrt(n) * dinvgauss(y, mu, lambda))
 else sigma <- 0.5/(sqrt(n) * dinvgauss(med, mu, lambda))
 cum.prob <- pinvgauss(med, mu, lambda)
 return(med, sigma, cum.prob)
}

qinvgauss<-
function(p, mu = stop("no mean arg"), lambda = 1)
{
 # Quantiles of the inverse Gaussian distribution
 # Dr Paul Bagshaw
 # Centre National d'Etudes des Telecommunications (DIH/DIPS)
 # Technopole Anticipa, France
 # paul.bagshaw@cnet.francetelecom.fr
```

```
 # 23 Dec 98
 #
 if(any(mu <= 0.)) stop("mu must be positive")
 if(any(lambda <= 0.))
        stop("lambda must be positive")
 n <- length(p)
 if(length(mu) > 1 && length(mu) != n)
        mu <- rep(mu, length = n)
 if(length(lambda) > 1 && length(lambda) != n)
        lambda <- rep(lambda, length = n)
 thi <- lambda/mu
 U <- qnorm(p)
 r1 <- 1 + U/sqrt(thi) + U^2/(2 * thi) + U^3/(8 * thi * sqrt(thi))
 x <- r1
 for(i in 1:10) {
        cum <- pinvgauss(x, 1., thi)
        dx <- (cum - p)/dinvgauss(x, 1., thi)
        dx <- ifelse(is.finite(dx), dx, ifelse(p > cum, -1, 1))
        dx[dx < -1] <- -1
        if(all(dx == 0.))
                break
        x <- x - dx
 }
 x * mu
}

dinvgauss<-
function(x, mu = stop("no shape arg"), lambda = 1)
{
 # Density of inverse Gaussian distribution
 # GKS 15 Jan 98
 #
 if(any(mu <= 0)) stop("mu must be positive")
 if(any(lambda <= 0))
        stop("lambda must be positive")
 d <- ifelse(x > 0, sqrt(lambda/(2 * pi * x^3)) * exp(( - lambda * (x - mu)^2)/(2 * mu^2 * x)),
        0)
 if(!is.null(Names <- names(x)))
        names(d) <- rep(Names, length = length(d))
 d
}

pinvgauss<-
function(q, mu = stop("no shape arg"), lambda = 1)
{
 # Inverse Gaussian distribution function
 # GKS 15 Jan 98
 #
 if(any(mu <= 0)) stop("mu must be positive")
 if(any(lambda <= 0))
        stop("lambda must be positive")
 n <- length(q)
 if(length(mu) > 1 && length(mu) != n)
        mu <- rep(mu, length = n)
 if(length(lambda) > 1 && length(lambda) != n)
        lambda <- rep(lambda, length = n)
```

```
 lq <- sqrt(lambda/q)
 qm <- q/mu
 p <- ifelse(q > 0, pnorm(lq * (qm - 1)) + exp((2 * lambda)/mu) * pnorm( - lq * (qm + 1)), 0)
 if(!is.null(Names <- names(q)))
          names(p) <- rep(Names, length = length(p))
 p
}

###################################

# Script for calling Fortran subroutine to get the
# global maximum likelihood

#########################################################################
# The Interface of S-Plus and Fortran subroutine          #
# This function providing the interface to          #
#  Fortran subroutines                        #
#                                  #
# User needs to provide the following variables:          #
# ndim: size of the data sig2                    #
# nobs: size of observable data                    #
# sizedata: Contain the size of related obs and sig          #
# sigma2: The correspondence sigma square of variable(sig) data    #
# obsdata: Observable data                    #
# minth: The lower boundary of the estimate range        #
# maxth: The upper boundary of the estimate range          #
# tablefile: The result output file designated by user     #
#########################################################################

# Created by Edward Chang, July 2004
nlm3b <- function(ndim,nobs,sizedata,sigma2,obsdata,minth, maxth, tablefile) {
  if(!loadlib()) loadlib()

  workpath <- getpwd()

  # The following lines assign value to the calling parameters
   mm <- ndim
   lg <- rep(0,mm)
   epsiln <- rep(0,mm)
   thetas <- rep(0,mm)
   values <- rep(0,mm)
   lhdghat<-0
   supports=0
   retiv <- rep(0,mm+60)
   flag=T

  # The following statement is calling Fortran subroutine
  results <- .Fortran( "dnlm3b_",
            ndim=as.integer(ndim),
            nobs=as.integer(nobs),
            sizedata=as.integer(sizedata),
            sigma2=as.double(sigma2),
            obsdata=as.double(obsdata),
            minth=as.double(minth),
            maxth=as.double(maxth),
            lhdghat=as.double(lhdghat),
```

```
            supports=as.integer(supports),
            lg=as.double(lg),
            epsiln=as.double(epsiln),
            thetas=as.double(thetas),
            values=as.double(values),
            iv=as.integer(retiv),
            flag=as.logical(flag)
          )

  # The following lines present the result to screen
   m <- results$supports
   cat( "******* The number of support points is :", m, "\n" )
   lhdghat <- results$lhdghat
   cat( "******* The maximum likelihood is :", lhdghat, "\n" )
   epsilon <- results$epsiln[1:m]
   cat( "******* The component proportions are :", epsilon, "\n" )
   thetas <- results$thetas[1:m]
   cat( "******* The estimated parameter values are :", thetas, "\n" )
   values <- results$values[1:m]
   cat( "******* The residual values are :", values, "\n" )
   flag <- results$flag

  # The following lines create the output result file named as tablefile.
   cat( ndim, nobs, minth, maxth, "\n", file = tablefile )
   cat( "log-like: ", lhdghat, "Num supports: ", m, "\n", file = tablefile, append=1 )
   cat( "   thetas       weights       values", "\n", file = tablefile, append=1 )
   for(i in 1:m) {
    cat(thetas[i],"\t",epsilon[i],"\t",values[i],"\n", file = tablefile, append=i>0 )
   }
}
#####################################################################


######
# Interface can be modified to accept the name of ydata and sizedata
#  The following function wrapes the above function and provide
#  user an easy interface. Users do not need to go further
#  to the interface details and only need to provide the following
#  five parameters
#
#  min: The lower boundary of the estimate range
#  max: The upper boundary of the estimate range
#  ydata: The filepath of the ydata
#  sizedata: The filepath of the sizedata
#  tablefile: The result output file designated by user
######
# Created by Edward Chang, July 2004
fortrancall <- function(min, max, ydata, sizedata, tablefile) {

 obs <- scan(ydata,what=numeric(),multi.line=F)
  nn<-length(obs)

  # data <- scan("D:\\Math-Soft\\Project\\S-Plus\\s-fortran\\size.dat",what=numeric())
 data <- scan(sizedata,what=numeric(),multi.line=F)
  n1<-length(data)
  n <-n1/2
```

```
  data1 <- matrix(data,nrow=n,ncol=2, byrow=T)
  sig <- data1[1:n,1]
  sizes <- data1[1:n,2]
  sig2 <- sig**2

  # Results: access the above interface
  test <- nlm3b(n,nn,sizes,sig2,obs,min,max,tablefile)
}

# Created by Edward Chang, July 2004
# Call the DOS command to make the copy
 dos.copy <- function(filename) {
  workpath <- getpwd()
  file1 <- paste(workpath, "\\npar.lst",sep="")
   file2 <- paste(workpath, "\\",filename,sep="")
   dos.copy.cmd <- paste("copy ", file1,file2)
   dos(dos.copy.cmd)
 }

data.prep<-function(filepath,flag.bad=c(3,5),shapeReg=.65,
            area2Per=.65,openPer=.3,ignoredRate=.25,
            bgdcFlag=1,fgdcFlag=1, saturatedV = 50000) {
# Created by Edward Chang, April 2005
# Data preparation : read the raw data file and create
#            the data frame for further use
# filepath : string indicated the absolute path of the data file
# the following arguments are quality factors
# shapeReg : Shape Regularity
# openPer  : Open Perimeter
# area2Per : Area-2-Perimeter
# ignoredRate : ignored pixel rate
# bgdcFlag : Background contaminated flag
# fgdcFlag : Foreground contaminated flag
# saturatedV : Indicate Saturated Value
#

  raw_read.table(filepath,sep="\t",header=T)
  raw_raw[order(raw$labels),]
  nlen_length(raw$yf)
  nvec_c(1:nlen)

  rawdata_cbind(raw, nvec)
  nn_length(names(rawdata))
  names(rawdata)[nn]_"no"

  boolqf_rawdata$shapeReg > shapeReg
  boolqf_boolqf & rawdata$area2Per > area2Per
  boolqf_boolqf & rawdata$openPer < openPer
  boolqf_boolqf & rawdata$ignoredArea/rawdata$spotArea < ignoredRate
  boolqf_boolqf & rawdata$bgdcFlag != bgdcFlag
  boolqf_boolqf & rawdata$fgdcFlag != fgdcFlag
  boolqf_boolqf & rawdata$xf < saturatedV
  boolqf_boolqf & rawdata$xb < saturatedV
  boolqf_boolqf & rawdata$yf < saturatedV
  boolqf_boolqf & rawdata$yb < saturatedV
  boolqf_boolqf & rawdata$yf != 0
```

```
  boolqf_boolqf & rawdata$yb != 0
  boolqf_boolqf & !is.element(rawdata$flag,flag.bad)


# When dealing with the ImaGene data, the following two lines
# should be runned to obtain the correct area
  rawdata$nf_round(rawdata$nf/9,digits=0)
  rawdata$nb_round(rawdata$nb/9,digits=0)
  rawdata$spotArea_round(rawdata$spotArea/9,digits=0)
  rawdata$ignoredArea_round(rawdata$ignoredArea/9,digits=0)

  nrawdata_rawdata[boolqf,]
  data.qf_nrawdata[,c(1:12,nn)]
  data.qf
  return( data.qf, rawdata )
}


# Created by Edward Chang, June 2005
# Turkey's method for removing the outliers
rm.outliers<-function(vec) {
  abc_vec
  qt1_quantile(abc,.25)
   qt3_quantile(abc,.75)
   iqr_(qt3-qt1)*1.5
#    iqr_(qt3-qt1)*2.0
   lbd_qt1-iqr
   ubd_qt3+iqr

   cbc_abc[abc>lbd & abc<ubd]
#    ma_mean(cbc)
   cbc_abc>lbd & abc<ubd
   return(cbc)
}

# Created by Edward Chang, June 2005
# Identify the spots which near saturated spot and get removed
# Public data version
pub.cls.blk<-function(vnblk,vblk,thresh=50000) {
 numblk<-length(vblk$flag)
    vsec_vblk$sector
 vr_vblk$row
 vc_vblk$column
 rmid_rep(T,numblk)

 for( i in 1:numblk ) {
          ii_vr[i]
          jj_vc[i]
       kk_vsec[i]
          if( ii - 1 < 0 ) {
                  vrow_c(ii,ii+1)
          } else if( ii + 1 > 32 ) {
                  vrow_c(ii-1,ii)
          } else {
                   vrow_c(ii-1,ii,ii+1)
          }
```

```
        if( jj - 1 < 0 ) {
                vcol_c(jj,jj+1)
        } else if( jj + 1 > 42 ) {
                vcol_c(jj-1,jj)
        } else {
                vcol_c(jj-1,jj,jj+1)
        }

        for( rk in vrow ) {
                for( ck in vcol ) {
                        vdata_vnblk[ vnblk$row==rk && vnblk$column==ck &&
vnblk$sector ==kk, ]
                        vdata
                        if( length(vdata$yf) > 0 ) {
                          if( (vdata$xf > thresh || vdata$xb > thresh) ) {
                                rmid[i]=F
                          }
                        }
                }
        }
 }
 return(rmid)
}

# Created by Edward Chang, June 2005
# Identify the spots which near saturated spot and get removed
# ImaGene Data Version
cls.blk<-function(vnblk,vblk,thresh=45000) {
 numblk<-length(vblk$flag)
 vr_vblk$row
 vc_vblk$column
 rmid_rep(T,numblk)

 for( i in 1:numblk ) {
        ii_vr[i]
        jj_vc[i]
        if( ii - 1 < 0 ) {
                vrow_c(ii,ii+1)
        } else if( ii + 1 > 32 ) {
                vrow_c(ii-1,ii)
        } else {
                vrow_c(ii-1,ii,ii+1)
        }

        if( jj - 1 < 0 ) {
                vcol_c(jj,jj+1)
        } else if( jj + 1 > 42 ) {
                vcol_c(jj-1,jj)
        } else {
          vcol_c(jj-1,jj,jj+1)
        }

        for( rk in vrow ) {
                for( ck in vcol ) {
                        vdata_vnblk[ vnblk$row==rk & vnblk$column==ck, ]
```

```
                                vdata
                                if( length(vdata$yf) > 0 ) {
                                  if( vdata$xf > thresh || vdata$xb > thresh ) {
                                        rmid[i]=F
                                  }
                                }
                        }
                }
        }
 return(rmid)
}

# Created by Edward Chang, April 2005
# Data preparation : read the raw data file and create
#            the data frame for further use
#            ( Version for public data)

pub.data.prep<-function(filepath,flag.bad=c(3,5),shapeReg=.65,area2Per=.65,openPer=.3,
      ignoredRate=.25,bgdcFlag=1,fgdcFlag=1, saturatedV = 50000) {
  rawdata_read.table(filepath,sep="\t",header=T)

#  vnblk_rawdata[rawdata$labels!="Blank",]
#  vblk_rawdata[rawdata$labels=="Blank",]
#  cc_pub.cls.blk(vnblk,vblk)
#  rawdata_rawdata[cc,]

  boolqf_ !is.element(rawdata$flag,flag.bad)
  boolqf_boolqf & rawdata$xf < saturatedV
  boolqf_boolqf & rawdata$xb < saturatedV
  boolqf_boolqf & rawdata$yf < saturatedV
  boolqf_boolqf & rawdata$yb < saturatedV

  nrawdata_rawdata[boolqf,]
  data.qf_nrawdata[,3:14]
  data.qf
  return( data.qf, rawdata )
}
data.oper<-function(dataset,flag.bad=c(3,5),saturatedV = 50000) {

  boolqf_ !is.element(dataset$flag,flag.bad)
  boolqf_boolqf & dataset$xf < saturatedV
  boolqf_boolqf & dataset$xb < saturatedV
  boolqf_boolqf & dataset$yf < saturatedV
  boolqf_boolqf & dataset$yb < saturatedV

  dataset_dataset[boolqf,]
  return(dataset)
}
################################################################
```

5.2 R script files

5.2.1 bgd_correction.r

```
#********** Step 1 - set up data and environment *************
# Start the program here !!

# Obtain current working directory, user designate the work path where
# the result file will be stored. User also can this script
# and backsub.ssc in this home directory
# The following function is to obtain the home directory
#
getpwd <- function() {
#  return("{Your working directory }")
#  Please change next line to return your work directory
      return("D:\\Math-Soft\\Project\\S-Plus\\s-fortran")
}

# Load the dynamic link library which includes the Fortran subroutine.
# User can put this *.dll anywhere, generally it will be put
# in the lib of splus which will be like
# {R Home directory}/lib/
loadlib<-function() {
#  dll.load("{R Home directory}\\lib\\r-dnlm3b.dll")
#  Please make necessary change to the next line with the format above.
    dyn.load("D:\\Math-Soft\\R\\rw1090\\lib\\r-dnlm3b.dll")
}

loadlib()

# Obtain the workpath and the data file name for y.dat and size.dat
 workpath <- getpwd()
 ydata <- paste(workpath, "\\y.dat",sep="")
 sizedata <- paste(workpath, "\\size.dat",sep="")

# User should designate the storate place of backsub.ssc here.
# Either you copy the file to a any place, saying "c:/Temp/s-plus/backsub.ssc"
#   and designate backsub.file<-" c:/Temp/s-plus/backsub.ssc "
# Or just copy it to your work directory and use the next line
backsub.file<-paste( workpath, "\\backsub.r", sep="")
source(file=backsub.file,local=F)
options( digits = 15 )

#************************************************************
# Create output and plots for work.dataset array 2 in paper

# Prepare dataset
# Either you can designate the file directory as the next line
# Or store your data in the data directory and use the next two lines
dataset.path<-"c:/temp/tmp/Caren-data/epa-b"
datafile<-paste(dataset.path, "/t31h24a16na.txt",sep="");

# Designate data name and work title to present in graphs
data.name_"t31h24a16na.lst"
work.title<-"epa-b: t31h24a16na"

############################################################
```

```
# Users needs only to change the directory and dll path
# above if they do not want to change the algotithms
##########################################################

# Get the raw data and clean the data
rawdata<-get.rawdata(datafile)
dataqf<-data.clean(rawdata)
blk<-dataqf[dataqf$labels == "Blank",]
nblk<-dataqf[dataqf$labels != "Blank",]

# If using background to estimate the distribution of background,
# blkindex should be set to "T" to let function call the backgrdsub()
blkindex <- T
work.dataset <- blk

length(work.dataset$yf)

### delete the spot near saturated spot.
cc<-cls.blk(nblk,blk)
work.dataset<-work.dataset[cc,]

# Set up the work database
work.dataset<-work.dataset[order(work.dataset$labels),]

#################### Remove outliers in blank spots
# Delete outliers in yf
abc<-work.dataset$yf
cbc<-rm.outliers(abc)
ma<-mean(abc[cbc])

work.dataset<-work.dataset[cbc,]

# Delete outlier in yb
abc<-work.dataset$yb
cbc<-rm.outliers(abc)
ma<-mean(abc[cbc])
work.dataset<-work.dataset[cbc,]

######## may not need to delete outliers for yf & yb
######## separately and do the following only
# delete outlier in yf-yb
abc<-work.dataset$yf-work.dataset$yb
hist(abc,main="Histogram of y_f-y_b before deleting outliers")
qqnorm(abc)
qqline(abc)
cbc<-rm.outliers(abc)
abc<-abc[cbc]
hist(abc,xlab="yf-yb",main="Histogram of yf-yb after deleting outliers")
ma<-mean(abc)
qqnorm(abc,main="Normal QQ plot for yf-yb")
qqline(abc)
work.dataset<-work.dataset[cbc,]
length(work.dataset$yf)

################## end of Remove outliers
```

```
# Designate the bad spot so that to get rid of them in the late analysis
bad.spot<-c(3,5)

#********** End of Step 1 - run to here ********************

#********** Step 2 - estimate G0 by blank spots **************
X<-max( work.dataset$yf )

#Checking distribution of yf for blank spots
summary(work.dataset$yf)
hist(work.dataset$yf)

# Get the upper boundary of the estimate range
xmaxtitle<-paste("X =", X)

if( blkindex ) {
  work.data.max<-backgrdsub(work.dataset,flag.bad=bad.spot,max.dif=X,
       mytitle=work.title, yfile=ydata, sizefile=sizedata)
} else {
  work.data.max<-backsub(work.dataset,flag.bad=bad.spot,max.dif=X,
      mytitle=work.title, yfile=ydata, sizefile=sizedata)
}

# run fortran programme
tablefile <- paste(workpath, "\\npar.lst",sep="")
fortrancall(0,X, ydata,sizedata,tablefile)

#filename <- paste(workpath, "\\dataset\\", data.name, sep="")
dos.copy(data.name)
#********** End of Step 2 **************************

#********** Start of Step 3: Background correction *******
# read in y_b for non-blank spots
# rawdata has most columns from Imagene dataset
 raw<-rawdata
raw.nonblank<-raw[raw$labels!="Blank",]
tmpraw<-data.oper(raw.nonblank,flag.bad=bad.spot)

# The following two lines set up the working dataset
work.dataset<-tmpraw
a<-work.dataset

X<-max( work.dataset$yb )
xmaxtitle<-paste("X =", X)
work.data.max<-backsub(work.dataset,flag.bad=bad.spot,
      max.dif=X,mytitle=work.title, yfile=ydata, sizefile=sizedata)
filename<-tablefile

# Compute the posterior
sm.max<-postest(work.data.max$ydif, work.data.max$c.sigma, work.data.max$c.sizes,
        work.data.max$data, tablefile=filename)

#Problem - why are there NA's in posterior?  Because outliers y_b 12,582, 12,226
b<-sm.max$post

# The following lines are to get rid of the NA spots
```

```
cna<-b[is.na(b)]
cc<-is.na(b)
ind.na<-(1:length(b))[cc]
cna<-b[is.infinite(b)]
cc<-is.infinite(b)
ind.infinite<-(1:length(b))[cc]
ind.na<-c(ind.na,ind.infinite)
a$yb[ind.na]
a$yf[ind.na]
length(a$yf)
yb<-a$yb
yf<-a$yf

# r.area<-a$nb/(a$nb+a$spotArea)

if( length(ind.na) > 0 ) {
   yb<-yb[-ind.na]
   yf<-yf[-ind.na]
   mub.post<-b[-ind.na]
   atmp<-a[-ind.na,]
} else {
   mub.post<-b
   atmp<-a
}

# Get the interaction of background and foreground
b.diff<-(yb-mub.post)
hist(b.diff, xlab="S", main="Histogram of S_hat")
summary(b.diff)

qqnorm(b.diff)
qqline(b.diff)
plot(yf, b.diff, main="Plot: S vs yf",
    ylab="S: yb-post(mu_b)", xlab="y_f")

# Plot of S_hat vs. yf
c1<-yf > 0
cltit<-"S_hat(=y_b - post) vs yf"

ind.reg<-(1:length(yf))[c1]
yf3<-yf[c1]
dif3<-b.diff[c1]
plot(yf3,dif3,main="Plot: S vs yf ",
   xlab="yf", ylab="S: yb - post(mu_b) ")
abline(0,0)

# Plot of S_hat vs. yf in log scale
cltit<-"Plot: S' vs log(yf)"
yf3<-log(yf[c1])
dif3<-(log(yb)-log(mub.post))[c1]

plot(yf3,dif3,main="Plot: S' vs log(yf) ",
   xlab="log(yf)", ylab="S'=log(yb)-log(post(mu_b))")
abline(0,0)

plot(yb, mub.post, main="y_b vs. est(y_b)" )
```

```
# Target: estimate of "true" foreground
mu.f<-yf+b.diff

# plot for the mu.f vs. yf
 plot(yf, mu.f, main="Plot: mu_f vs y_f",ylab="mu_f",xlab="y_f")
abline(0,1)

# Write back the estimate "true" value to dataset with column name "mu.f"
tmpbgcor<-cbind(atmp, mu.f)
nn<-length(names(tmpbgcor))
names(tmpbgcor)[nn]<-"mu.f"

nvec<-rep(NA,length(raw$yf))
bgcor<-cbind(raw,nvec)
nn<-length(names(bgcor))
names(bgcor)[nn]<-"mu.f"
bgcor[tmpbgcor$no,]$mu.f<-tmpbgcor$mu.f

# Cleanup
# The following two lines are the output for further study.
out.bgcor<-bgcor
out.muf<-mu.f

# Clean up
 alist<-objects()
rm(list=alist)
rm(alist)
#************ End of Step 3 ************************
##########################################################
```

5.2.2 backsub.r

It the same file as backsub.ssc