Lab: Using lumi Package to Process Illumina Microaray Data

Pan Du[1]*, Simon Lin[1]†

August 5, 2007

[1]Robert H. Lurie Comprehensive Cancer Center
Northwestern University, Chicago, IL, 60611, USA

Contents

1 Introduction

In this Lab, we will provide a work flow of a typical use case, as shown in Figure 1. We will explain each step and provide examples. We select the Barnes data set as the example data for this Lab session. The Barnes data set measured a dilution series of two human tissues, blood and placenta. It includes six samples with the titration ratio of blood and placenta as 100:0, 95:5, 75:25, 50:50, 25:75 and 0:100. The samples were hybridized on HumanRef-8 BeadChip (Illumina,

*dupan@northwestern.edu
†s-lin2@northwestern.edu

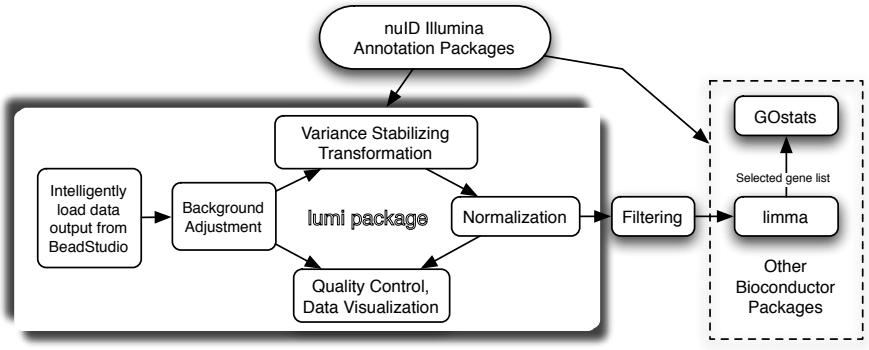

Figure 1: Processing flow chart of the lumi package

Inc) in duplicate. See (Barnes, et al., 2005) for details. The Barnes data has been packaged as lumiBarnes data package at the Bioconductor Experiment Data web page.

1.1 Required packages and installation

This Lab requires the users to install packages: *lumi* (>= 1.3.21), *vsn*, *limma* and *lumiBarnes* (>= 1.3.2) (Experiment Data package). First, we need to load these packages:

```
> library(lumi)
> library(vsn)
> library(limma)
> library(lumiBarnes)
> set.seed(0xbadbeef)
```

2 Description of work flow steps

2.1 Input data

The lumiR function can "intelligently" read different versions of BeadStudio output files. It also allows the expert users to customize the input data, like selecting the columns they want to input. Here, we just load the lumiBarnes data for convenience. Users can play with their own data following the help information of lumiR function.

```
> ## Load the Barnes data set
> data("lumiBarnes")
> ## Only selected the dilution samples
> selChip = !is.na(lumiBarnes$pctBlood)
> x.lumi = lumiBarnes[, selChip]
> ## summary of the data
> x.lumi
```

```
Summary of BeadStudio output:
        Illumina Inc. BeadStudio version 1.4.0.1
        Normalization = none
        Array Content = 11188230_100CP_MAGE-ML.XML
        Error Model = none
        DateTime = 2/3/2005 3:21 PM
        Local Settings = en-US

Major Operation History:
            submitted            finished                                                comma
1 2007-03-25 02:23:16 2007-03-25 02:25:22          lumiR("../data/Barnes_gene_profile.txt
2 2007-03-25 02:25:27 2007-03-25 02:25:29                                lumiQ(x.lumi = x.lum
3 2007-03-25 02:26:00 2007-03-25 02:26:06 addNuId2lumi(x.lumi = x.lumi, lib = "lumiHumanV1
4 2007-03-25 02:30:28 2007-03-25 02:30:28                             Subsetting 21966 feature
5 2007-08-05 17:56:54 2007-08-05 17:56:55                                 Subsetting 12 sample

Object Information:
LumiBatch (storageMode: lockedEnvironment)
assayData: 21966 features, 12 samples
  element names: beadNum, detection, exprs, se.exprs
phenoData
  rowNames: A01, A02, ..., B06 (12 total)
  varLabels and varMetadata:
    sampleID: The unique Illumina microarray Id
    label: The label of the sample
    ...: ...
    replicate: technique replicate
    (5 total)
featureData
  rowNames: ZpF0SBA8lTA9BF6Ku4, fl97V6._QueT9ZYg1k, ..., W3oFhJHVApHUkUN4XU (21966 total)
  varLabels and varMetadata:
    TargetID: The Illumina microarray identifier
    TargetID: The number of detectable measurements of the gene
experimentData: use 'experimentData(object)'
Annotation [1] "lumiHumanV1"
```

2.2 Background removal

Due to the random distribution of the beads on the surface, the background removal is usually simpler than Affy or other microarray platforms. Usually it just subtracts an offset, estimated based on the negative control probes. We recommend the BeadStudio output the background corrected data (without normalization). The lumiB function can also do background correction if the data includes the control probe information.

```
> ## Since the Barnes data was not background removed, we will do background adjustment fi
> ## The background estimation will be based on the control probe information.
> x.lumi = lumiB(x.lumi, method='bgAdjust', probs=0)
```

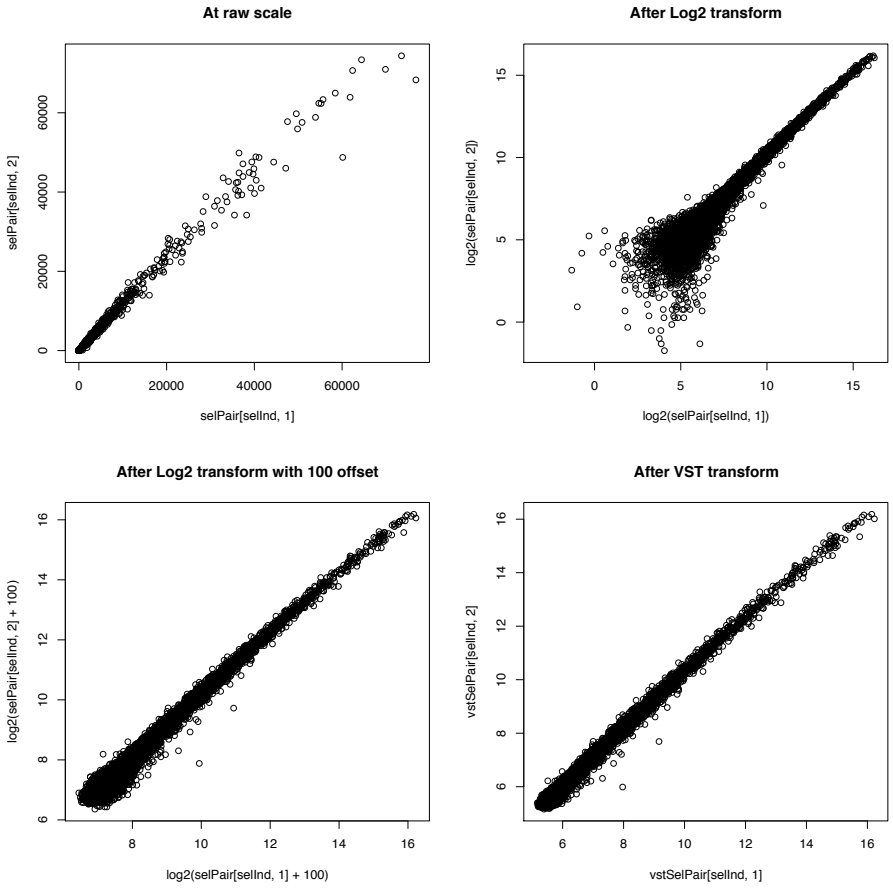

Figure 2: Comparing scatter plots at raw scale and log2 scale

2.3 Variance stabilization

Variance stabilization is critical for the subsequent statistical inference to identify differential genes from microarray data. We devised a variance-stabilizing transformation (VST) by taking advantages of larger number of technical replicates available on the Illumina microarray.

Why we need variance stabilization? Figure 2 shows the scatter plots of a technical replicate at raw and log2 scale. We can see the data has big variance in the range of high amplitudes. In order to stabilize the variance, a common practice is to add a log 2 transformation to the data. However, the log2 transformation will increase the variance in the range of low amplitude, especially when the data is background adjusted, as shown in the upper right plot of Figure 2. Manually add an offset can somehow stabilize the variance, like shown in the lower left plot of Figure 2. However, in reality we do not know how much offset need to be added. The VST is a generalized logarithm transformation, it can automatically estimate the offset and other parameters of the transformation by fitting the mean and standard deviation relations, and successfully stabilize the variance.

2.4 Normalization

The function lumiN provides options for different normalization algorithms, like, RSN, quantile, loess and vsn. The default method is RSN (Robust Spline Normalization). The RSN method combines the good features of quantile normalization (rank invariant) and curving fitting normalization (continuous mapping function). Here we just want to show a potential problem of quantile normalization people usually ignored, i.e., the quantile normalization may remove small but important difference during normalization and is unrecoverable.

```
> # Demonstrate the limitation of quantile noramlization
> # Let us simulate three arrays in a matrix: each row is a gene; each column is an array.
> #  Say, they are biological replicates, similar to each other,
> #  but still have critical differences. The biological variation is
> #  necessary to keep.
> #
>  x <- as.matrix(data.frame(
+      x1= seq (1, 10000) + rnorm (10000,0,0.3),
+      x2= seq (1, 10000) + rnorm (10000,0,0.3),
+      x3= seq (1, 10000) + rnorm (10000,0,0.3)
+ ))
> print(x[1:5,])

          x1        x2       x3
[1,] 0.909882 0.9244702 1.302719
[2,] 1.841455 1.7396976 1.604793
[3,] 3.215119 3.0307992 2.728285
[4,] 3.582006 3.8882589 4.067494
[5,] 4.777974 4.8481935 4.674779

> #
> # Now, after quantile normalization, it seems that some of the
> # variations are gone.
> x1 <- normalize.quantiles(x)
> print(x1[1:5,])

         [,1]     [,2]     [,3]
[1,] 1.045690 1.045690 1.045690
[2,] 1.728648 1.728648 1.728648
[3,] 2.991401 2.991401 2.991401
[4,] 3.845919 3.845919 3.845919
[5,] 4.766982 4.766982 4.766982

> #
> # To confirm this, we take a look at the variance of all genes,
> # before and after normalization.
> v <- apply(x, 1, var)
> summary(v)

     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
3.111e-06 2.558e-02 6.189e-02 8.919e-02 1.232e-01 9.695e-01
```

```
> v1 <- apply(x1, 1, var)
> summary(v1)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00000 0.00000 0.00000 0.01042 0.00000 0.82690
```

It demonstrates that quantile normalization can excessively remove critical variations that are intrinsic to the biology. Using the same simulated data, we test the RSN normalization.

```
> x.lumiN <- lumiN(x)

2007-08-05 17:57:19 , processing array  1
2007-08-05 17:57:20 , processing array  2
2007-08-05 17:57:20 , processing array  3

> print(x.lumiN[1:5,])

            x1        x2       x3
[1,] 0.9127386 0.9244702 1.285342
[2,] 1.8463381 1.7396976 1.584440
[3,] 3.2224272 3.0307992 2.698207
[4,] 3.5898885 3.8882589 4.027644
[5,] 4.7875792 4.8481935 4.630951

> v.lumiN <- apply(x.lumiN, 1, var)
> summary(v.lumiN)

     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
0.0000144 0.0396400 0.0982600 0.1401000 0.1947000 1.7790000
```

2.5 Filtering

In the microarray analysis of whole genome chips, usually the majority of the probes have very low amplitudes and are not measurable across all samples. These probes will be removed before further analysis. Here we will check the the correlation between technical replicates of these probes. Figure 3 shows the histogram of the correlation coefficients. We can see the correlation coefficients of the probes with no sample "Present" is close to Gaussian, while the probes with all sample present with higher correlation coefficients.

```
> ## Estimate the detection count of lumiBarnes data
> presentCount <- detectionCall(x.lumi)
> ## If using old version (before 1.3.2) lumiBarnes library
> # presentCount <- detectionCall(x.lumi, 0.99)
> dataMatrix <- exprs(x.lumi)
> #
> ## check the correlation between 6 pairs of technical replicates
> cc.bad <- apply(dataMatrix[presentCount == 0,], 1, function(x) cor(x[1:6], x[7:12]))
> cc.good <- apply(dataMatrix[presentCount == ncol(dataMatrix),], 1, function(x) cor(x[1:6
```

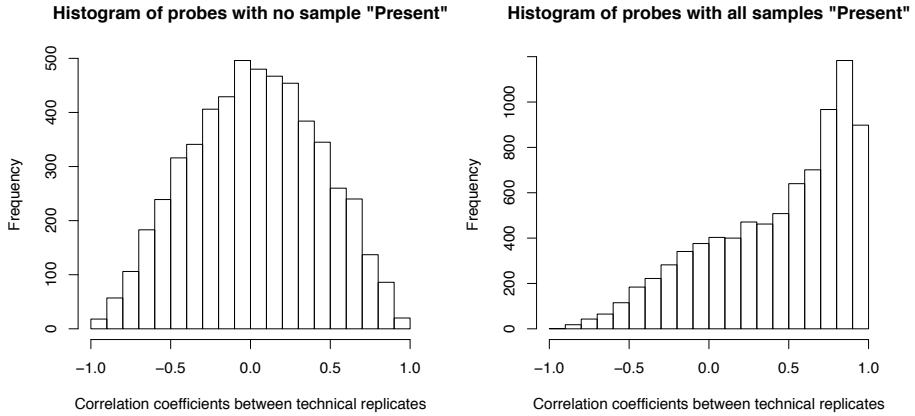

Figure 3: Comparing histograms of technical replicate correlation of probes with no samples "Present" and all samples "Present"

2.6 Annotation with nuID

Here just show some examples of how nuID works (Du, P., Kebbe, W.A, Lin, S.M., Biology Direct, 2:16, 2007).

```
> ## examples of nuID
> nuID <- featureNames(x.lumi)[1]
> print(nuID)

[1] "ZpF0SBA8lTA9BF6Ku4"

> ## convert to nucleotide sequence
> probeSeq <- id2seq(nuID)
> print(probeSeq)

[1] "GGCACCTCACAGAACAAATTAGCCCATAAATTCAACACCTGGAGGGTGTG"

> ## convert back to nuIDs
> nuID.new <- sapply(probeSeq, seq2id)
> print(nuID.new)

GGCACCTCACAGAACAAATTAGCCCATAAATTCAACACCTGGAGGGTGTG
                              "ZpF0SBA8lTA9BF6Ku4"

> ## check a random sequence
> is.nuID(nuID)

[1] TRUE

> ## check a random sequence
> is.nuID('adfqeqe')

[1] FALSE
```

3 Evaluation of the VST algorithm

Next, we will evaluate the VST algorithm by comparing with Log2 and VSN algorithms by processing the Barnes data set.

3.1 Preprocessing

Load the data and subset the arrays of interest:

```
> ## load the library
> library("lumi")
> library("vsn")
> library("limma")
> library("lumiBarnes")
> set.seed(0xbadbeef)
> #
> ## Load the Barnes data set
> data("lumiBarnes")
> selChip = !is.na(lumiBarnes$pctBlood)
> x.lumi = lumiBarnes[, selChip]
> #
> # Background removal
> x.lumi = lumiB(x.lumi, method='bgAdjust', probs=0)
> #
> ## VST transform
> x.lumi.vst <- lumiT(x.lumi)
> #
> ## Quantile normalization
> x.lumi.vst.quantile <- lumiN(x.lumi.vst, method='quantile')
> #
> ## We can also use lumiExpresso to combine multiple preprocessing steps
> ## log2 transform and Quantile normalization
> x.lumi.log.quantile <- lumiExpresso(x.lumi, varianceStabilize.param=list(method='log2'),
> #
> ## VSN normalization: use lts.quantile=0.5 since in the blood/placenta
> ##  comparison more genes are differentially expressed than what is
> ##   expected by the default of 0.9.
> x.lumi.vsn <- lumiExpresso(x.lumi, variance.stabilize=FALSE, normalize.param=list(method
> #
> ## combine them as a list
> normDataList <- list('VST-Quantile'=exprs(x.lumi.vst.quantile),
+                      'Log2-Quantile'=exprs(x.lumi.log.quantile),
+                     'VSN'=exprs(x.lumi.vsn))
```

3.2 Correlation between the technical replicate microarrays

A good preprocessing method will improve the correlation between the technical replicate microarrays. Here will calculate the correlation between six pairs of technical replicate chips and plot them as the box plot, as shown in Figure 4. We can see VST improves the consistency between replicates.

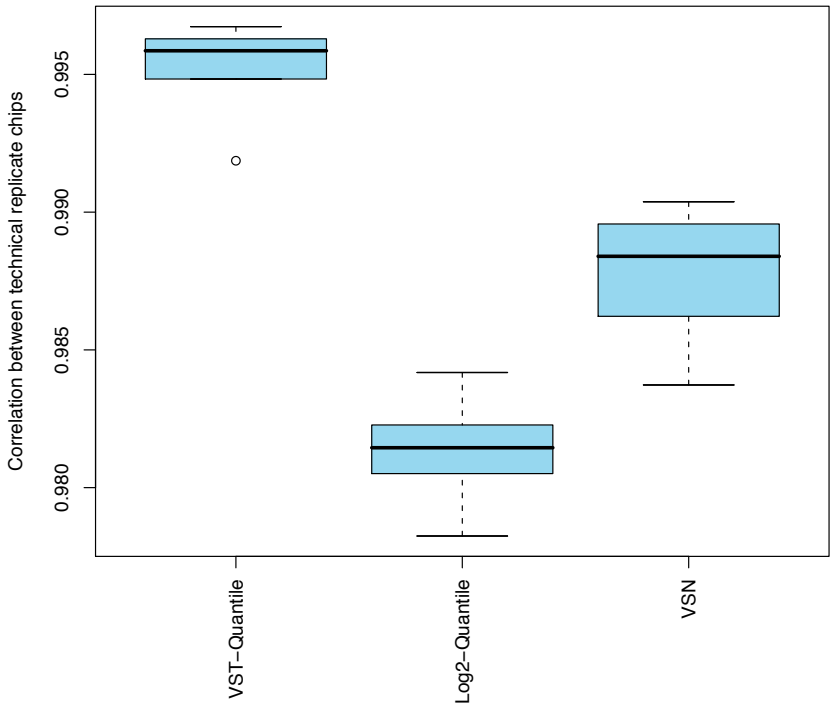

Figure 4: Correlation between technical replicate chips after preprocessing

```
> ## replicate index
> repl1 <- 1:6; repl2 <- 7:12
> ## Check the correlation between technical replicates
> chipCorList = matrix(as.numeric(NA), nrow=length(repl1), ncol=length(normDataList))
> colnames(chipCorList)= names(normDataList)
> for (i in seq(along=normDataList))
+   for (j in seq(along=repl1))
+     chipCorList[j,i] = cor(normDataList[[i]][, c(repl1[j], repl2[j])])[1,2]
```

3.3 Variance stabilizing between the technical replicates

A good variance stabilizing method should stabilize the variance between the technical replicates. Here we plot the mean and standard deviation relations between a pair of technical replicates, as shown in Figure 5. Users can select other pairs of replicates and plot the pictures.

```
> ## select the technical replicates
> selChip <-  c(repl1[1],repl2[1])
> oldpar <- par(mfrow=c(length(normDataList),1))
> for (i in 1:length(normDataList)) {
+   meanSdPlot(normDataList[[i]][, selChip], ylab='Standard deviation',
+       main=names(normDataList)[i], ylim=c(0,1))
+ }
> par(oldpar)
```

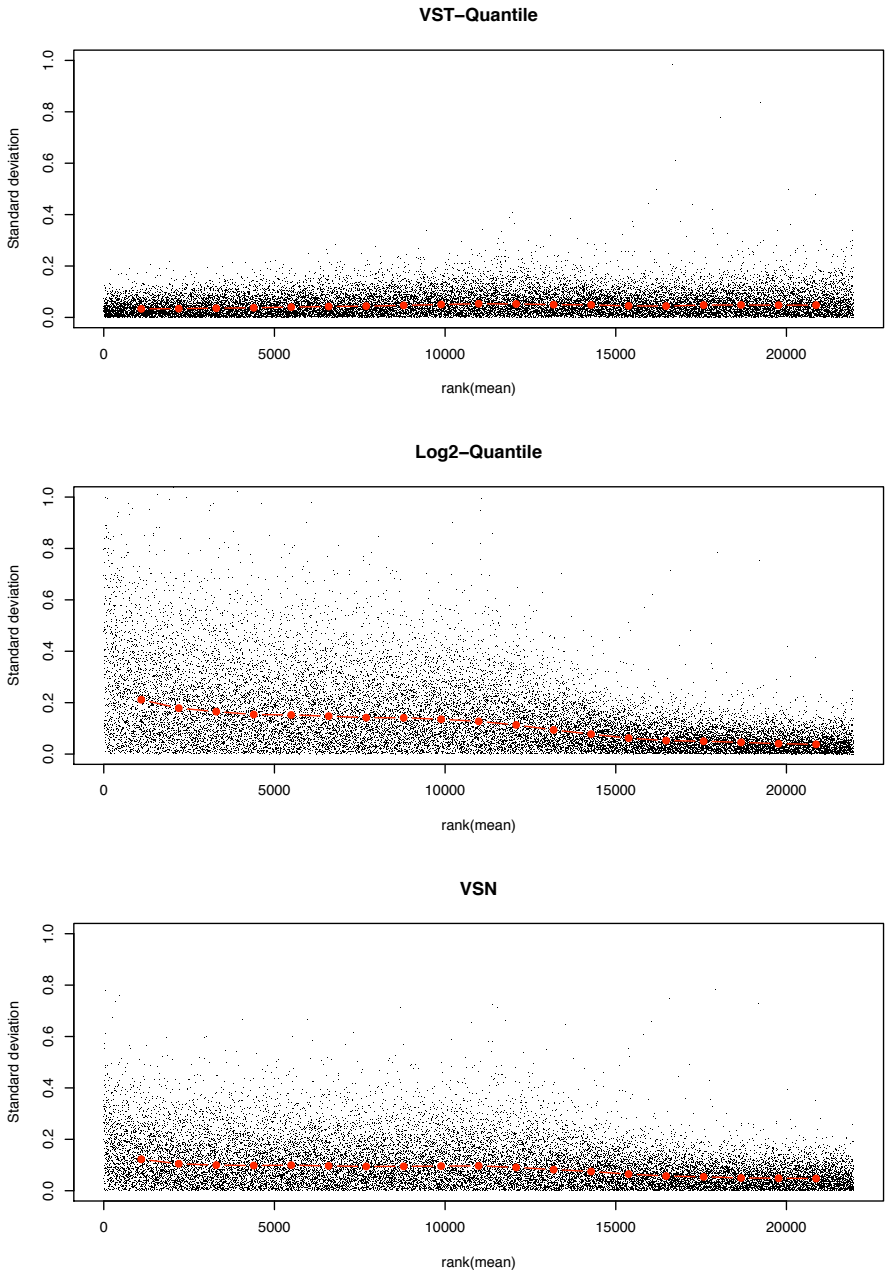

Figure 5: Mean and standard deviation relations of the technical replicate microarrays A01 and B01.

3.4 Evaluation based on the identification of differentially expressed genes

We evaluated the methods based on the concordance of normalized intensity profile and real dilution profile of the selected probes. Following Barnes et al. (2005), we defined a concordant probe as a signal from a probe with a correlation coefficient larger than 0.8 between the normalized intensity profile and the real dilution profile (six dilution ratios with two replicates at each dilution). The method in *limma* package selects differential expressed genes . To better evaluate the overall performance, we first ranked the probes with their p-values from low to high, then calculate the percentage of concordant probes among different number of top probes, as shown in Figure 6. The result indicates that VST results outperforms Log2.Quantile in terms of the concordance evaluation.

Identify the differentially expressed genes by using limma package:

```
> sampleInfo <- pData(phenoData(x.lumi))
> sampleType <- paste(sampleInfo[,'pctBlood'], sampleInfo[,'pctPlacenta'], sep=':')
> sampleType <- paste('c', sampleType, sep='')
> presentCount <- detectionCall(x.lumi)
> ## If using old version (before 1.3.2) lumiBarnes library
> # presentCount <- detectionCall(x.lumi, 0.99)
> ## Comparing index
> compareInd <- c(repl1[1:2], repl2[1:2])
> # compareInd <- c(1,6,9,14)        ## additional example
> compareType <- sampleType[compareInd]
> fitList <- NULL
> for (i in 1:length(normDataList)) {
+         selDataMatrix <- normDataList[[i]]
+         selDataMatrix <- selDataMatrix[presentCount > 0, ]
+         selProbe <- rownames(selDataMatrix)
+         compareMatrix <- selDataMatrix[, compareInd]
+
+         design <- model.matrix(~ 0 + as.factor(compareType))
+         colnames(design) <- c('A', 'B')
+         fit1 <- lmFit(compareMatrix, design)
+         contMatrix <- makeContrasts('A-B'=A - B, levels=design)
+         fit2 <- contrasts.fit(fit1, contMatrix)
+         fit <- eBayes(fit2)
+         fitList <- c(fitList, list(fit))
+ }
> names(fitList) <- names(normDataList)
```

The following code estimates the number of concordance genes (a probe with a correlation coefficient larger than 0.8 between the normalized intensity profile and the real dilution profile (six dilution ratios with two replicates at each dilution)) among the top differentially expressed genes (ranked based on p-values estimated by *limma*).:

```
> ## Check the correlation of the top differentiated genes based on the limma results
> ## rank the genes based on the p-values of limma result
> topNumList <- c(30, seq(35, 500, by=30))
```

```
> corTh <- 0.8
> corrList <- NULL
> highCorrNumMatrix <- NULL
> for (i in 1:length(fitList)) {
+         probeList <- rownames(fitList[[i]]$p.value)
+         fc.i <- fitList[[i]]$coef[,1]
+         ordProbe.i <- probeList[order(abs(fitList[[i]]$p.value[,1]), decreasing=F)]
+
+         selDataMatrix <- normDataList[[i]][ordProbe.i, ]
+
+         modelProfile1 <- c(100, 95, 75, 50, 25, 0, 100, 95, 75, 50, 25, 0)
+         profileMatrix <- selDataMatrix
+         corr1 <- apply(profileMatrix, 1, cor, y=modelProfile1)
+         names(corr1) <- ordProbe.i
+         matchNum.j <- NULL
+         for (topNum.j in topNumList) {
+                 topProbe.j <- ordProbe.i[1:topNum.j]
+                 matchNum.j <- c(matchNum.j, length(which(abs(corr1[topProbe.j]) > corTh)
+         }
+         highCorrNumMatrix <- cbind(highCorrNumMatrix, matchNum.j)
+         corrList <- c(corrList, list(c(list(corr1))))
+ }
> rownames(highCorrNumMatrix) <- topNumList
> colnames(highCorrNumMatrix) <- names(corrList) <- names(fitList)
```

4 Session Info

```
> toLatex(sessionInfo())
```

- R version 2.5.1 Patched (2007-07-11 r42199), `powerpc-apple-darwin8.9.1`
- Locale: `en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, grDevices, graphics, methods, stats, tools, utils
- Other packages: Biobase 1.14.0, affy 1.14.1, affyio 1.4.0, annotate 1.14.1, limma 2.10.5, lumi 1.3.22, lumiBarnes 1.3.2, mgcv 1.3-25, vsn 2.2.0

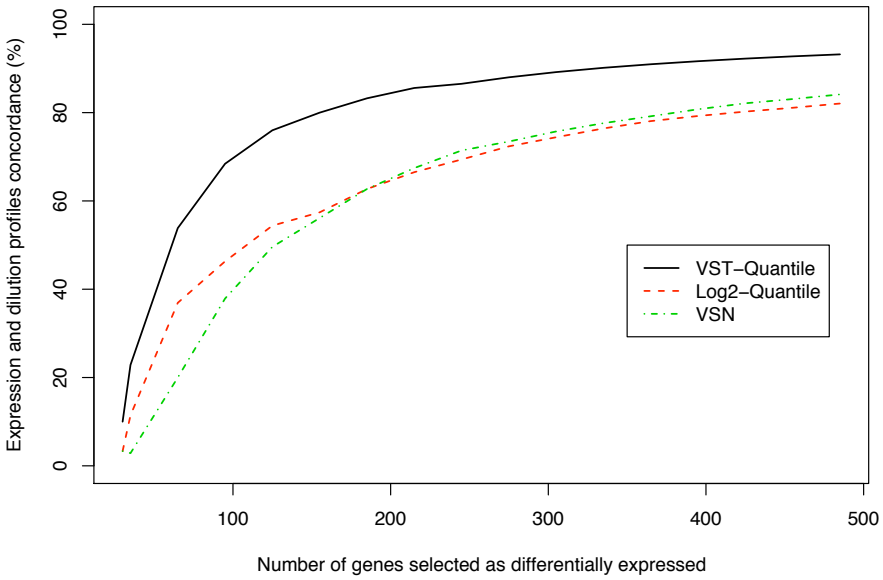

Figure 6: The concordance between the expression and dilution profiles of the selected differentially expressed genes