

Evaluation of VST algorithm in lumi package

Pan Du^{1*}, Simon Lin^{1†}, Wolfgang Huber^{2‡}, Warren A. Kibbe^{1§}

August 25, 2007

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

²EBI/EMBL, Cambridge, UK

Contents

1	Introduction	1
2	Required packages and data preprocessing	2
3	Evaluation of the VST algorithm	3
3.1	Correlation between the technical replicate microarrays	3
3.2	Variance stabilizing between the technique replicate microarrays	3
3.3	Variation within replicates vs. variation between conditions . . .	3
3.4	Correlation between the expression profiles and dilution profile .	6
3.5	Evaluation based on the identification of differentially expressed genes	8
4	Conclusion	10
5	Session Info	11
6	Reference	12

1 Introduction

Variance stabilization is critical for the subsequent statistical inference to identify differentially expressed 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. Here we use the Barnes data set, which has been packaged as lumiBarnes data package at the Bioconductor Experiment Data web page, to evaluate the VST algorithm. We will compare VST with popular base-2 logarithm transform and VSN method. To

*dupan@northwestern.edu

†s-lin2@northwestern.edu

‡huber@ebi.ac.uk

§wakibbe@northwestern.edu

facilitate the comparison, we used popular quantile normalization for both VST and log2 transformed data.

2 Required packages and data preprocessing

The evaluation requires the users to install packages: *lumi*, *vsn*, *genefilter*, *limma* and *lumiBarnes* (Experiment Data package). First, we need to load these packages:

```
> # library("lumi")
> library("vsn")
> library("genefilter")
> library("genefilter")
> library("RColorBrewer")
> library("limma")
> library("lumiBarnes")
> set.seed(0xbadbeef)
```

Load the data and subset the arrays of interest:

```
> ## Load the Barnes data set
> data("lumiBarnes")
> selChip = !is.na(lumiBarnes$pctBlood)
> x.lumi <- lumiBarnes[, selChip]
> presentCount <- detectionCall(x.lumi)
> ## Since the Barnes data was not background removed, we will do background adjustment fi
> ## The background estimation will be based on the control probe information.
> ## As the old version lumiBarnes library does not include controlData slot, we will che
> if (nrow(x.lumi@controlData) == 0) {
+     ## We will use the control probe information in the example.lumi in the updated
+     data(example.lumi)
+     x.lumi@controlData <- example.lumi@controlData
+ }
> x.lumi <- lumiB(x.lumi, method='bgAdjust')
> repl1 <- which(x.lumi$replicate=="A")
> repl2 <- which(x.lumi$replicate=="B")
> stopifnot(sum(selChip)==12L, length(repl1)==6L, length(repl2)==6L)
```

Preprocess:

```
> ## VST transform and Quantile normalization
> x.lumi.vst <- lumiT(x.lumi)
> x.lumi.vst.quantile <- lumiN(x.lumi.vst, method='quantile')
> ## log2 transform and Quantile normalization
> x.lumi.log <- lumiT(x.lumi, method='log2')
> x.lumi.log.quantile <- lumiN(x.lumi.log, method='quantile')
> ## 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 <- lumiN(x.lumi, method='vsn', lts.quantile=0.5)
> ## Add the vsn based on technical replicates
```

```

> vsn.pair <- exprs(x.lumi)
> cor.i <- NULL
> for(i in 1:length(repl1)) {
+     vsn.pair[, c(i, i+length(repl1))] <- exprs(vsn2(vsn.pair[, c(repl1[i], repl2[i]))
+ }
> # vsn.quantile <- normalize.quantiles(vsn.pair)
> # rownames(vsn.quantile) <- rownames(vsn.pair)
> # colnames(vsn.quantile) <- colnames(vsn.pair)
>
>
> normDataList <- list('VST-Quantile'=exprs(x.lumi.vst.quantile),
+                      'Log2-Quantile'=exprs(x.lumi.log.quantile),
+                      'VSN'=exprs(x.lumi.vsn)) # , 'VSN-Quantile'=vsn.quantil

```

3 Evaluation of the VST algorithm

3.1 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 1. We can see VST...

```

> ## Check the correlation between technique replicates
> tempDataList <- c(normDataList, list(vsn.pair))
> names(tempDataList) <- c(names(normDataList), 'VSN.pairwise')
> chipCorList <- matrix(as.numeric(NA), nrow=length(repl1), ncol=length(tempDataList))
> colnames(chipCorList) <- names(tempDataList)
> for (i in seq(along= tempDataList))
+   for (j in seq(along=repl1))
+     chipCorList[j,i] = cor(tempDataList[[i]][, c(repl1[j], repl2[j])])[1,2]

```

3.2 Variance stabilizing between the technique replicate microarrays

A good variance stabilizing method should stabilize the variance between the technique replicates. Here we plot the mean and standard deviation relations between a pair of technique replicates, as shown in Figure 2. Users can select other pairs of replicates and plot the pictures.

3.3 Variation within replicates vs. variation between conditions

To assess the signal to noise ratio, we assess

$$\frac{\sigma_{\text{between groups}}^2}{\sigma_{\text{within groups}}^2}.$$

For n groups, by its generalisation, the F -statistic.

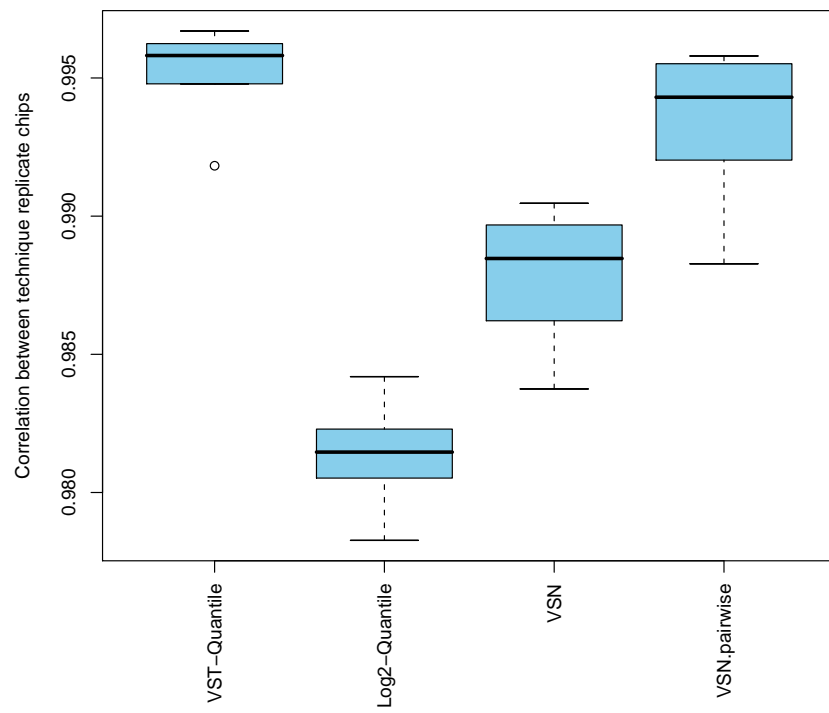


Figure 1: Correlation between technique replicate chips after preprocessing

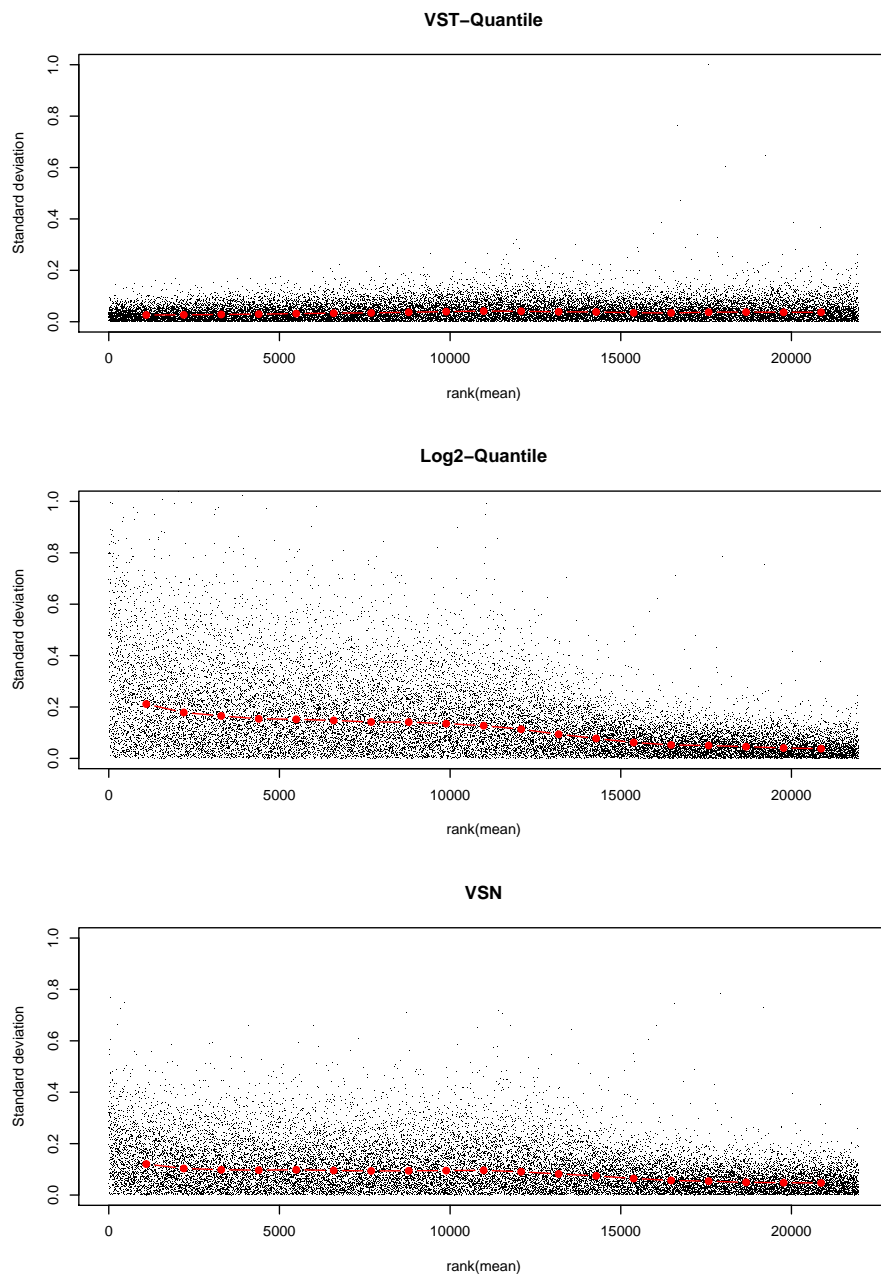


Figure 2: Mean and standard deviation relations of the technical replicate microarrays A01 and B01.

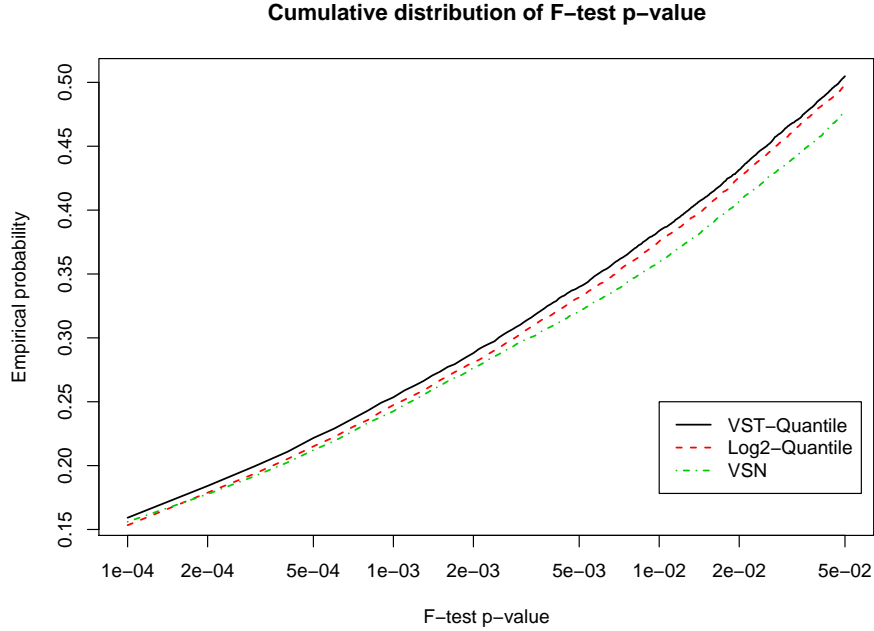


Figure 3: Cumulative distribution functions of p -values obtained from a) reporter-wise F -tests (by factor `fac`). These are monotonous measures of the ratio between variation within replicates and variation between conditions, or in other words, the signal-to-noise ratio.

```
> fac <- factor(paste(x.lumi$pctBlood, x.lumi$pctPlacenta, sep=":"))
> rf <- lapply(normDataList, function(x) rowFtests(x[presentCount > 0,], fac=fac))
> ef <- sapply( rf, function(x) ecdf(x$p.value))
```

The result is shown in Figure 3. We can see the difference among these methods are not big, however, the VST is consistently better than the log2 and VSN methods.

3.4 Correlation between the expression profiles and dilution profile

Here we want to compare the correlation between the expression profiles and dilution profile. Because these concordant genes are more likely to be related with the dilution process, a good transformation should improve or at least not worsen the correlation of the expression profiles and dilution profile. Figure 4 shows, VST transformed data improve this correlation because there are more probes with high correlation (the absolute values of correlation coefficient close to 1).

```
> modelProfile1 <- c(100, 95, 75, 50, 25, 0, 100, 95, 75, 50, 25, 0)
> corrList <- lapply(normDataList, function(x) {
+   x <- x[presentCount > 0, ]
```

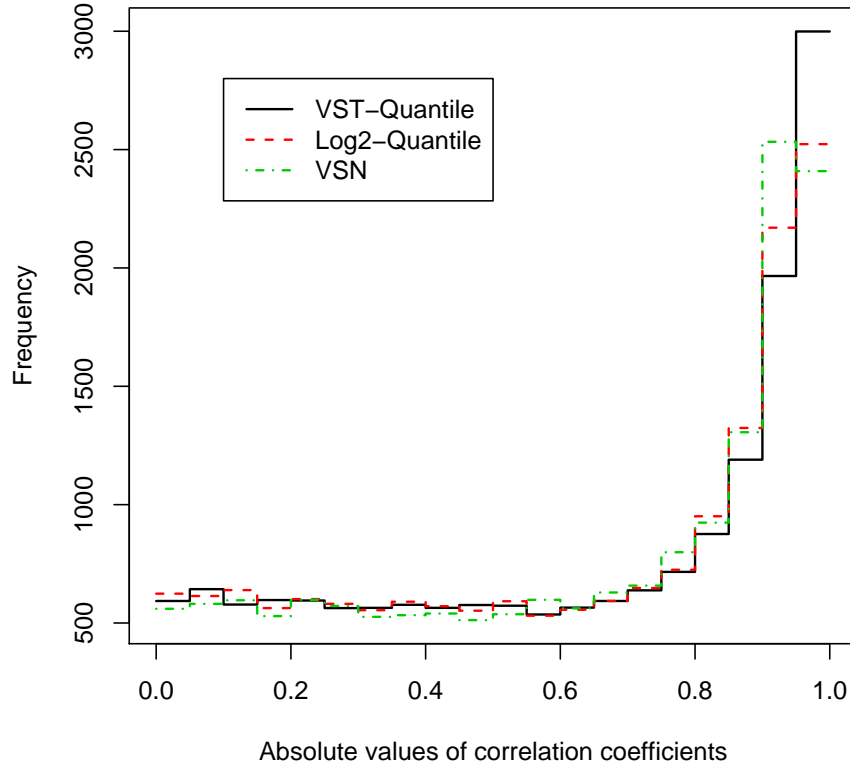


Figure 4: Compare the histogram of the correlation between the expression profiles and dilution profile

```
+      cor1 <- apply(x, 1, cor, y=modelProfile1)
+      return(cor1)
+    } )
```

3.5 Evaluation based on the identification of differentially expressed genes

For better evaluation, we want to evaluate the VST algorithm based on the detection of differentially expressed genes. First, we want to see the percentage of concordant probes (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 probes (ranking based on F-test p-values). The result is shown in Figure 5. We can see the VST processed data has obviously higher percentage of concordant probes than the log2 and VSN methods.

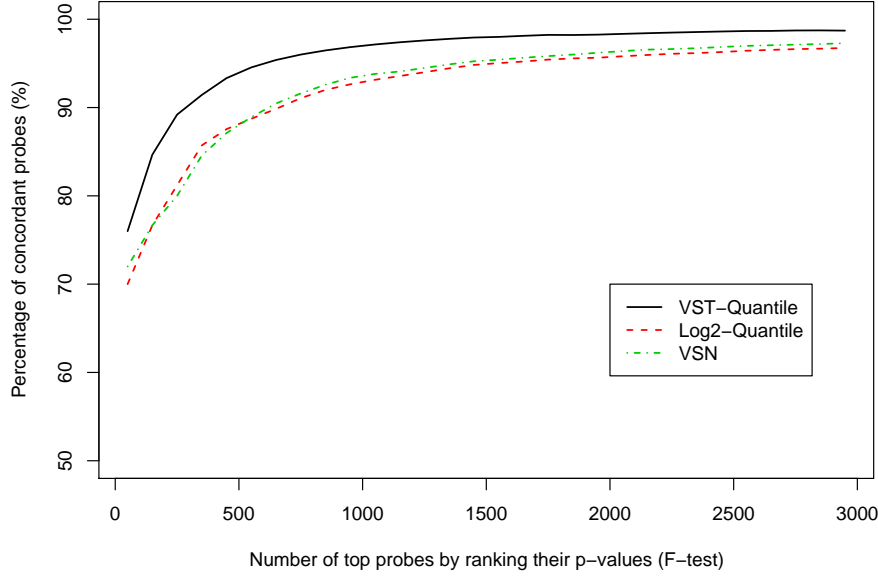


Figure 5: Cumulative distribution functions of p -values obtained from a) reporter-wise F -tests (by factor **fac**). These are monotonous measures of the ratio between variation within replicates and variation between conditions, or in other words, the signal-to-noise ratio.

```

> topNumList <- seq(50, 3000, by=100)
> corTh <- 0.8
> highCorrNumMatrix <- NULL
> for (i in 1:length(rf)) {
+   probeList <- names(rf[[i]]$p.value)
+   ordProbe.i <- probeList[order(abs(rf[[i]]$p.value), decreasing=FALSE)]
+   corr1 <- corrList[[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)
+ }
> rownames(highCorrNumMatrix) <- topNumList
> colnames(highCorrNumMatrix) <- names(rf)

```

The result is shown in Figure 5. We can see the difference among these methods are not big, however, the VST is consistently better than the log2 and VSN methods.

Next, we selected the differentially expressed genes by comparing two conditions. The p -values will be estimated by the Bioconductor *limma* package. 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-quantile outperforms Log2.Quantile in terms of the concordance evaluation.

Identify the differentially expressed genes by using limma package:

```
> ## Select the comparing chip index
> sampleInfo <- pData(phenoData(x.lumi))
> sampleType <- paste(sampleInfo[, 'pctBlood'], sampleInfo[, 'pctPlacenta'], sep=':')
> sampleType <- paste('c', sampleType, sep='')
> ## Comparing index
> ## used in the paper (the most challenging comparison):
> compareInd <- c(repl1[1:2], repl2[1:2])
> compareType <- sampleType[compareInd]
> fitList.limma <- 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.limma <- c(fitList.limma, list(fit))
+ }
> names(fitList.limma) <- names(normDataList)
```

Estimate the number of concordance probes (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 probes based on the limma results
> ## rank the probes based on the p-values of limma result
> fitList <- fitList.limma
> topNumList <- c(30, seq(35, 1000, by=30))
> corTh <- 0.8
> highCorrNumMatrix <- NULL
> for (i in 1:length(fitList)) {
+   probeList <- rownames(fitList[[i]]$p.value)
+   ordProbe.i <- probeList[order(abs(fitList[[i]]$p.value[,1]), decreasing=FALSE)]
+   profileMatrix <- normDataList[[i]][ordProbe.i, ]
+
+   modelProfile1 <- c(100, 95, 75, 50, 25, 0, 100, 95, 75, 50, 25, 0)
+   cor1 <- apply(profileMatrix, 1, cor, y=modelProfile1)
```

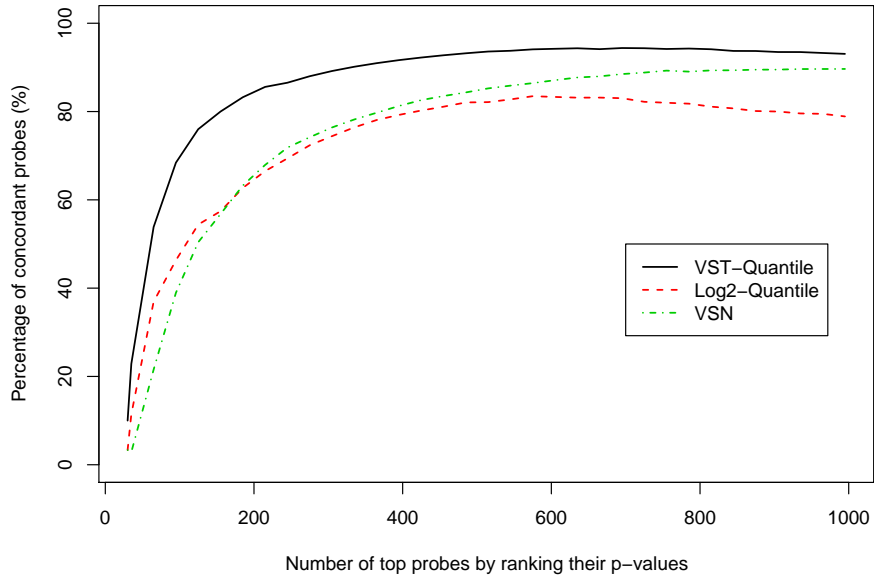


Figure 6: The concordance between the expression and dilution profiles of the selected differentially expressed genes

```

+      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)
+ }
> rownames(highCorrNumMatrix) <- topNumList
> colnames(highCorrNumMatrix) <- names(fitList)

```

4 Conclusion

The users can select different samples for the comparison and change the cut-off thresholds in the evaluation. The results should be similar, i.e., the VST algorithm is better than the log2 transformation and VSN for this evaluation data set because it utilizes the mean and standard deviation information at the bead-level.

5 Session Info

```

> toLatex(sessionInfo())

```

- R version 2.6.0 Under development (unstable) (2007-08-04 r42421), powerpc-apple-darwin8.10.1
- 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, splines, stats, tools, utils
- Other packages: AnnotationDbi 0.0.88, Biobase 1.15.23, DBI 0.2-3, RColorBrewer 1.0-1, RSQLite 0.6-2, affy 1.15.7, affyio 1.5.6, annotate 1.15.3, genefilter 1.15.9, geneplotter 1.15.4, lattice 0.16-2, limma 2.11.9, lumi 1.3.25, lumiBarnes 1.3.2, mgcv 1.3-25, preprocessCore 0.99.12, survival 2.32, vsn 3.0.11
- Loaded via a namespace (and not attached): KernSmooth 2.22-21, grid 2.6.0

6 Reference

1. Lin, S.M., Du, P., Kibbe, W.A., "Model-based Variance-stabilizing Transformation for Illumina Mi-croarray Data", under review
2. Barnes, M., Freudenberg, J., Thompson, S., Aronow, B. and Pavlidis, P. (2005) "Experimental comparison and cross-validation of the Affymetrix and Illumina gene expression analysis platforms", Nucleic Acids Res, 33, 5914-5923.