Gene Selection Using *GeneSelectMMD*

Jarrett Morrow
remdj@channing.harvard.edu,
Weilianq Qiu
stwxq@channing.harvard.edu,
Wenqing He
whe@stats.uwo.ca,
Xiaogang Wang
stevenw@mathstat.yorku.ca,
Ross Lazarus
ross.lazarus@gmail.com,

October 17, 2016

1 Introduction

This document demonstrates how to use the *GeneSelectMMD* package to detect significant genes and to estimate false discovery rate (FDR), false non-discovery rate (FNDR), false positive rate (FPR), and false negative rate (FNR), for a real microarray data set based on the method proposed by Qiu et al. (2008). It also illustrates how to visualize the fit of the model proposed in Qiu et al. (2008) to the real microarray data set when the marginal correlations among subjects are zero or close to zero.

The *GeneSelectMMD* package is suitable for the case where there are only two tissue types and for the case where tissue samples within a tissue type are conditionally independent given the gene (c.f. Qiu et al., 2008).

Note that starting from version 1.7.1, we reparameterize the model parameters. The input and output are the same as the previous versions. To improve speed, we use numerical optimization, instead of iteration based on explicit formula, to obtain parameter estimates. Since verions 2.0.7, the speed has also been improved by using Fortran 77 code for some of the less efficient operations. Lastly, in version 2.0.8 onward, the likelihood values provided to the user are based on the observed data, while in previous versions the likelihood was an expected value.

2 Methods

MMD assumes that the marginal distribution of a gene profile is a mixture of 3-component multivariate normal distributions with special structure for mean vectors and covariance matrices. The 3 component distributions correspond to 3 gene clusters: (1) cluster of genes over-expressed in the case group; (2) cluster of genes non-differentially expressed; and (3) cluster of genes under-expressed in the case group. Specifically, the marginal density of a gene profile is assumed to be

$$\begin{aligned} &f(\boldsymbol{x}|\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) = \pi_1 f_1(\boldsymbol{x}|\boldsymbol{\theta}_1) + \pi_2 f_2(\boldsymbol{x}|\boldsymbol{\theta}_2) + \pi_3 f_3(\boldsymbol{x}|\boldsymbol{\theta}_3), \\ &\pi_1 + \pi_2 + \pi_3 = 1, \pi_i > 0, i = 1, 2, 3, \end{aligned} \tag{1}$$

where π_1, π_2, π_3 are mixture proportions. The $m \times 1$ vector $\boldsymbol{x}$ is a realization of the random vector $\boldsymbol{X}$ that represents the transformed gene profile for a randomly selected gene over m tissue samples ($m = m_c + m_n$,

where m_c is the number of case tissue samples and m_n control tissue samples); $\boldsymbol{\theta}_k$, is the parameter set for the k-th component distribution f_k, $k = 1, 2, 3$; and f_1, f_2, and f_3 are the density functions for multivariate Normal distributions with the mean vectors

$$\boldsymbol{\mu}_1 = \begin{pmatrix} \mu_{c1}\mathbf{1}_{m_c} \\ \mu_{n1}\mathbf{1}_{m_n} \end{pmatrix}, \quad \boldsymbol{\mu}_2 = \mu_2\mathbf{1}_m, \quad \boldsymbol{\mu}_3 = \begin{pmatrix} \mu_{c3}\mathbf{1}_{m_c} \\ \mu_{n3}\mathbf{1}_{m_n} \end{pmatrix}. \tag{2}$$

and covariance matrices

$$\boldsymbol{\Sigma}_1 = \begin{pmatrix} \sigma^2_{c_1}\boldsymbol{R}_{c_1} & \mathbf{0} \\ \mathbf{0} & \sigma^2_{n_1}\boldsymbol{R}_{n_1} \end{pmatrix}, \quad \boldsymbol{\Sigma}_2 = \sigma^2_2\boldsymbol{R}_2, \quad \boldsymbol{\Sigma}_3 = \begin{pmatrix} \sigma^2_{c_3}\boldsymbol{R}_{c_3} & \mathbf{0} \\ \mathbf{0} & \sigma^2_{n_3}\boldsymbol{R}_{n_3} \end{pmatrix}, \tag{3}$$

respectively, where correlation matrix

$$\boldsymbol{R}_t = (1-\rho_t)\left[\boldsymbol{I}_{n_t} + \frac{\rho_t}{(1-\rho_t)}\mathbf{1}_{n_t}\mathbf{1}^T_{n_t}\right], \tag{4}$$

$t = c_1$, n_1, 2, c_3, or n_3. $n_t = m_c$ if $t = c_1$ or c_3; $n_t = m$ if $t = 2$; $n_t = m_n$ if $t = n_1$, or n_3. Here we assume, without loss of generality, that the first m_c elements of the random vector $\mathbf{X}$ are for the case tissue samples and the remaining m_n elements are for the control tissue samples. Let $\boldsymbol{\theta}_1 = (\mu_{c_1}, \sigma^2_{c_1}, \rho_{c_1}, \mu_{n_1}, \sigma^2_{n_1}, \rho_{n_1})^T$, $\boldsymbol{\theta}_2 = (\mu_2, \sigma^2_2, \rho_2)^T$, $\boldsymbol{\theta}_3 = (\mu_{c_3}, \sigma^2_{c_3}, \rho_{c_3}, \mu_{n_3}, \sigma^2_{n_3}, \rho_{n_3})^T$. Note that $\mu_{c_1} > \mu_{n_1}$ for component 1 in which genes are over-expressed in case tissue samples, and $\mu_{c_3} < \mu_{n_3}$ for component 3 where genes are under-expressed in case samples. Our prior belief is that the majority of genes are usually non-differentially expressed, so we assume $\pi_2 > \pi_1$ and $\pi_2 > \pi_3$.

The model parameters can be estimated using the EM algorithm (Dempster et al., 1977).

The cluster membership of a gene is determined by the posterior probability that the gene belongs to a cluster given its gene profile. The posterior probability is a function of the 3 component marginal density functions and the mixing proportions:

$$\begin{aligned} Pr(\text{gene } i \in \text{cluster } k|\boldsymbol{x}_i) =& \frac{\pi_k f_k(\boldsymbol{x}_i|\boldsymbol{\theta}_k)}{\pi_1 f_1(\boldsymbol{x}_i|\boldsymbol{\theta}_1) + \pi_2 f_2(\boldsymbol{x}_i|\boldsymbol{\theta}_2) + \pi_3 f_3(\boldsymbol{x}_i|\boldsymbol{\theta}_3)}, \\ k =& 1, 2, 3, \end{aligned} \tag{5}$$

Specifically, a gene is assigned to the a gene cluster if the posterior probability that the gene belongs to that gene cluster given its gene profile is larger than the posterior probability that the gene belongs to other gene clusters given its gene profile:

$$k_0 = \arg\max_{k=1,2,3} Pr(\text{gene } i \in \text{cluster } k|\boldsymbol{x}_i). \tag{6}$$

An important task is to assess the performance of a gene selection method so that different methods can be objectively compared. To evaluate the performance of a gene selection method, investigators usually compare the error rates, such as FDR, FNDR, FPR, and FNR, via simulation studies. However, when analyzing a real microarray data set, investigators are more interested in what the values of FDR, FNDR, FPR, and FNR are for this specific real microarray set. It is challenging to estimate FDR, FNDR, FPR, and FNR for real microarray data sets since the true gene cluster membership is unknown for real data sets. However, model-based gene clustering methods, such as Bayesian hierarchical models and MMD, can provide such estimates since these error rates can be expressed as functions of marginal density functions and mixing proportions, where the model parameters and mixing proportions can be estimated from the real microarray data. It is easy to use MMD to estimate the four error rates since MMD describes the distributions of gene expression levels directly via the marginal distributions, while it is usually difficult to derive the marginal density functions for Bayesian hierarchical models.

It is of practical importance to evaluate if a model fits a real microarray data set well. If a model does not fit well for the real microarray data set, then it makes no sense to estimate the error rates based on the model. Although it is quite challenging to asses the goodness of fit for multivariate data, especially

for non-normal multivariate data, it is possible to do so for some special cases. For example, when tissue samples are marginally independent, we could pool the gene expression levels across tissue samples for each type of tissue samples since they are all independent. We then could impose the theoretical density curve on the histogram of the pooled expression levels for each type of tissue samples. The parameters ρ_{c1}, ρ_{n1}, ρ_2, ρ_{c3}, and ρ_{n3} indicates the marginal correlations among tissue samples. Assuming that the marginal correlations are ignorable, we then could produce such a plot to evaluate the goodness of fit of MMD to the real microarray data set.

3 Reparametrization

To guarantee the constraints $\mu_{c_1} > \mu_{n_1}$ and $\mu_{c_3} < \mu_{n_3}$ in the iteration of the EM algorithm, we re-parameterize the model parameters:

$$\begin{aligned}\mu_{n_1} =&\mu_{c_1} - \exp(\triangle_{n_1}),\\ \mu_{n_3} =&\mu_{c_3} + \exp(\triangle_{n_3}).\end{aligned}$$

To make sure the marginal covariance matrices are positive definite, we set the constraints:

$$\begin{aligned}-\frac{1}{n_c-1} <&\rho_{c_1} < 1\\ -\frac{1}{n_n-1} <&\rho_{n_1} < 1\\ -\frac{1}{n-1} <&\rho_2 < 1\\ -\frac{1}{n_c-1} <&\rho_{c_3} < 1\\ -\frac{1}{n_n-1} <&\rho_{n_3} < 1\end{aligned}$$

To make sure the above inequalities hold, we reparametrize ρs as follows:

$$\begin{aligned}\rho_{c_1} =&\frac{\exp(r_{c_1}) - 1/(n_c-1)}{1+\exp(r_{c_1})}\\ \rho_{n_1} =&\frac{\exp(r_{n_1}) - 1/(n_n-1)}{1+\exp(r_{n_1})}\\ \rho_2 =&\frac{\exp(r_2) - 1/(n-1)}{1+\exp(r_2)}\\ \rho_{c_3} =&\frac{\exp(r_{c_3}) - 1/(n_c-3)}{1+\exp(r_{c_3})}\\ \rho_{n_3} =&\frac{\exp(r_{n_3}) - 1/(n_n-3)}{1+\exp(r_{n_3})}.\end{aligned}$$

Another set of constraints are:

$$\begin{aligned}0 <&\pi_k < 1, \quad k = 1,2,3,\\ \pi_3 =&1-\pi_1-\pi_2.\end{aligned}$$

Then the reparametrized MMD model is

$$f(\boldsymbol{x}) = \pi_1 f_1(\boldsymbol{x}|\boldsymbol{\theta}_1) + \pi_2 f_2(\boldsymbol{x}|\boldsymbol{\theta}_2) + (1-\pi_1-\pi_2) f_3(\boldsymbol{x}|\boldsymbol{\theta}_3), \tag{7}$$

where

$$\begin{aligned}\boldsymbol{\theta}_1 =&(\mu_{c_1}, \sigma^2_{c_1}, r_{c_1}, \triangle_{n_1}, \sigma^2_{n_1}, r_{n_1})^T,\\ \boldsymbol{\theta}_2 =&(\mu_2, \sigma^2_2, r_2)^T,\\ \boldsymbol{\theta}_3 =&(\mu_{c_3}, \sigma^2_{c_3}, r_{c_3}, \triangle_{n_3}, \sigma^2_{n_3}, r_{n_3})^T,\end{aligned} \tag{8}$$

and f_1, f_2, and f_3 are the density functions for multivariate normal distributions with mean vectors

$$\boldsymbol{\mu}_1 = \begin{pmatrix} \mu_{c_1}\mathbf{1}_{n_c} \\ [\mu_{c_1} - \exp(\triangle_{n_1})]\,\mathbf{1}_{n_n} \end{pmatrix},\ \boldsymbol{\mu}_2 = \mu_2\mathbf{1}_n,\ \boldsymbol{\mu}_3 = \begin{pmatrix} \mu_{c_3}\mathbf{1}_{n_c} \\ [\mu_{c_3} + \exp(\triangle_{n_3})]\,\mathbf{1}_{n_n} \end{pmatrix} \tag{9}$$

and covariance matrices

$$\begin{aligned} \boldsymbol{\Sigma}_1 =& \begin{pmatrix} \sigma^2_{c_1}\frac{n_c}{(n_c-1)(1+\exp(r_{c_1}))}\boldsymbol{R}_{0,c_1} & \mathbf{0} \\ \mathbf{0} & \sigma^2_{n_1}\frac{n_n}{(n_n-1)(1+\exp(r_{n_1}))}\boldsymbol{R}_{0,n_1} \end{pmatrix}, \\ \boldsymbol{\Sigma_2} =& \sigma^2_2\frac{n}{(n-1)(1+\exp(r_2))}\boldsymbol{R}_{0,2}, \\ \boldsymbol{\Sigma}_3 =& \begin{pmatrix} \sigma^2_{c_3}\frac{n_c}{(n_c-1)(1+\exp(r_{c_3}))}\boldsymbol{R}_{0,c_3} & \mathbf{0} \\ \mathbf{0} & \sigma^2_{n_3}\frac{n_n}{(n_n-1)(1+\exp(r_{n_3}))}\boldsymbol{R}_{0,n_3} \end{pmatrix}. \end{aligned} \tag{10}$$

The matrices $\boldsymbol{R}_{0,c_1}$, $\boldsymbol{R}_{0,n_1}$, $\boldsymbol{R}_{0,2}$, $\boldsymbol{R}_{0,c_3}$, and $\boldsymbol{R}_{0,n_3}$ are defined as below:

$$\begin{aligned} \boldsymbol{R}_{0,c_1} =& \boldsymbol{I}_{n_c} + \frac{(n_c-1)\exp(r_{c_1})-1}{n_{c_1}}\mathbf{1}_{n_c}\mathbf{1}^T_{n_c}, \\ \boldsymbol{R}_{0,n_1} =& \boldsymbol{I}_{n_n} + \frac{(n_n-1)\exp(r_{n_1})-1}{n_{n_1}}\mathbf{1}_{n_n}\mathbf{1}^T_{n_n}, \\ \boldsymbol{R}_{0,2} =& \boldsymbol{I}_n + \frac{(n-1)\exp(r_2)-1}{n_2}\mathbf{1}_n\mathbf{1}^T_n, \\ \boldsymbol{R}_{0,c_3} =& \boldsymbol{I}_{n_c} + \frac{(n_c-1)\exp(r_{c_3})-1}{n_{c_3}}\mathbf{1}_{n_c}\mathbf{1}^T_{n_c}, \\ \boldsymbol{R}_{0,n_3} =& \boldsymbol{I}_{n_n} + \frac{(n_n-1)\exp(r_{n_3})-1}{n_{n_3}}\mathbf{1}_{n_n}\mathbf{1}^T_{n_n}. \end{aligned} \tag{11}$$

4 Gene selection via *GeneSelectMMD*

The *GeneSelectMMD* includes four functions for gene selection: `gsMMD`, `gsMMD.default`, `gsMMD2`, and `gsMMD2.default`.

The functions `gsMMD` and `gsMMD2` accept the object derived from the class of *Bioconductor*'s `ExpressionSet` as data input, while the functions `gsMMD.default` and `gsMMD2.default` accept data matrix as data input.

The functions `gsMMD` and `gsMMD.default` will provide initial 3-cluster gene partitions (cluster of genes over-expressed in case group, cluster of genes non-differentially expressed, and cluster of genes under-expressed in case group) based on the gene-wise two-sample t-test or two-sample Wilcoxon rank-sum test. In situations where the user would like to provide the initial 3-cluster partition other than that provided by the gene-wise two-sample t-test or two sample Wilcoxon rank-sum test, the functions `gsMMD2` and `gsMMD2.default` could be used.

The rows of the input data matrix (or the data matrix derived from the object derived from the class `ExpressionSet`) are genes, while the columns are tissue samples. The tissue type of a tissue sample is indicated by the argument `memSubjects`, which is a $m \times 1$ vector, $m = m_c + m_n$, m_c is the number of tissue samples in the case group, and m_n is the number of tissue samples in the control group. `memSubjects[j]=1` indicates the j-th tissue sample is from the case group. `memSubjects[j]=0` indicates the j-th tissue sample is from the control group.

The output of gene selection functions include `dat`, `memGenes`, `memGenes2`, and `para`.

- `dat` is a data matrix with the same dimensions as the input data matrix. If no data transformation is performed, `dat` is the same as the input data matrix. Otherwise, it will be the transformed input data matrix.

- `memGenes` is a $G \times 1$ vector indicating the gene cluster membership, where G is the number of genes (i.e., the number of rows of the input data matrix). `memGenes[g]=1` indicates the g-th gene is assigned to the cluster of genes over-expressed in case group; `memGenes[g]=2` indicates the g-th gene is assigned to the cluster of genes non-differentially-expressed; `memGenes[g]=3` indicates the g-th gene is assigned to the cluster of genes under-expressed in case group.

- `memGenes2` is a variant of `memGenes`. `memGenes2[g]=1` means the g-th gene is differentially expressed, while `memGenes2[g]=0` means the g-th gene is non-differentially expressed, $g = 1, \ldots, G$.

- `para` is a 18×1 vector of parameters (π_1, π_2, π_3, μ_{c1}, σ^2_{c1}, ρ_{c1}, μ_{n1}, σ^2_{n1}, ρ_{n1}, μ_2, σ^2_2, ρ_2, μ_{c3}, σ^2_{c3}, ρ_{c3}, μ_{n3}, σ^2_{n3}, ρ_{n3},) for the model described in Equations (**??**)- (**??**), which can be estimated by the EM algorithm. `para[1]`, `para[2]`, and `para[3]` are the cluster proportions for the 3 gene clusters (over-expressed, non-differentially expressed, and under-expressed). `para[4]`, `para[5]`, and `para[6]` are the marginal mean, variance, and correlation of gene expression levels of cluster 1 (over-expressed genes) for case subjects; `para[7]`, `para[8]`, and `para[9]` are the marginal mean, variance, and correlation of gene expression levels of cluster 1 (over-expressed genes) for control subjects; `para[10]`, `para[11]`, and `para[12]` are the marginal mean, variance, and correlation of gene expression levels of cluster 2 (non-differentially expressed genes); `para[13]`, `para[14]`, and `para[15]` are the marginal mean, variance, and correlation of gene expression levels of cluster 3 (under-expressed genes) for case subjects; `para[16]`, `para[17]`, and `para[18]` are the marginal mean, variance, and correlation of gene expression levels of cluster 3 (under-expressed genes) for control subjects.

  Note that genes in cluster 2 are non-differentially expressed across case and control tissue samples. Hence there are only 3 parameters for cluster 2.

For example, to obtain differentially expressed genes for the ALL data (Chiaretti et al., 2004), we can run either

```
>   library(GeneSelectMMD)
>   library(ALL)
>   data(ALL)
>   eSet1 <- ALL[1:100, ALL$BT == "B3" | ALL$BT == "T2"]
>   mem.str <- as.character(eSet1$BT)
>   nSubjects <- length(mem.str)
>   memSubjects <- rep(0,nSubjects)
>   # B3 coded as 0, T2 coded as 1
>   memSubjects[mem.str == "T2"] <- 1
>   obj.gsMMD <- gsMMD(eSet1, memSubjects, transformFlag = TRUE,
+     transformMethod = "boxcox", scaleFlag = TRUE, quiet = TRUE)

Programming is running. Please be patient...
Programming is running. Please be patient...

>   para <- obj.gsMMD$para
>   print(round(para, 3))

     pi.1      pi.2      pi.3     mu.c1 sigma2.c1    rho.c1     mu.n1 sigma2.n1
    0.056     0.896     0.048     0.557     0.559    -0.069    -0.390     0.905
   rho.n1      mu.2  sigma2.2     rho.2     mu.c3 sigma2.c3    rho.c3     mu.n3
   -0.047    -0.002     0.981    -0.027    -0.845     0.280     0.035     0.569
sigma2.n3    rho.n3
    0.615    -0.039

>
```

or

```
>   library(GeneSelectMMD)
>   library(ALL)
>   data(ALL)
>   eSet1 <- ALL[1:100, ALL$BT == "B3" | ALL$BT == "T2"]
>   mat <- exprs(eSet1)
>   mem.str <- as.character(eSet1$BT)
>   nSubjects <- length(mem.str)
>   memSubjects <- rep(0,nSubjects)
>   # B3 coded as 0, T2 coded as 1
>   memSubjects[mem.str == "T2"] <- 1
>   obj.gsMMD <- gsMMD.default(mat, memSubjects, iniGeneMethod = "Ttest",
+           transformFlag = TRUE, transformMethod = "boxcox", scaleFlag = TRUE,
+           quiet=TRUE)
>   para <- obj.gsMMD$para
>   print(round(para, 3))
>
```

If we would like to provide the initial 3-cluster partition via the two sample Wilcoxon rank-sum test, then we can run either

```
>   library(GeneSelectMMD)
>   library(ALL)
>   data(ALL)
>   eSet1 <- ALL[1:100, ALL$BT == "B3" | ALL$BT == "T2"]
>   mem.str <- as.character(eSet1$BT)
>   nSubjects <- length(mem.str)
>   memSubjects <- rep(0,nSubjects)
>   # B3 coded as 0, T2 coded as 1
>   memSubjects[mem.str == "T2"] <- 1
>   myWilcox <-
+   function(x, memSubjects, alpha = 0.05)
+   {
+     xc <- x[memSubjects == 1]
+     xn <- x[memSubjects == 0]
+
+     m <- sum(memSubjects == 1)
+     res <- wilcox.test(x = xc, y = xn, conf.level = 1 - alpha)
+     res2 <- c(res$p.value, res$statistic - m * (m + 1) / 2)
+     names(res2) <- c("p.value", "statistic")
+
+     return(res2)
+   }
>   mat <- exprs(eSet1)
>   tmp <- t(apply(mat, 1, myWilcox, memSubjects = memSubjects))
>   colnames(tmp) <- c("p.value", "statistic")
>   memIni <- rep(2, nrow(mat))
>   memIni[tmp[, 1] < 0.05 & tmp[, 2] > 0] <- 1
>   memIni[tmp[, 1] < 0.05 & tmp[, 2] < 0] <- 3
>   print(table(memIni))
>   obj.gsMMD <- gsMMD2(eSet1, memSubjects, memIni, transformFlag = TRUE,
+        transformMethod = "boxcox", scaleFlag = TRUE, quiet = TRUE)
```

```
>   para <- obj.gsMMD$para
>   print(round(para, 3))
>
```

or

```
>   library(GeneSelectMMD)
>   library(ALL)
>   data(ALL)
>   eSet1 <- ALL[1:100, ALL$BT == "B3" | ALL$BT == "T2"]
>   mat <- exprs(eSet1)
>   mem.str <- as.character(eSet1$BT)
>   nSubjects <- length(mem.str)
>   memSubjects <- rep(0,nSubjects)
>   # B3 coded as 0, T2 coded as 1
>   memSubjects[mem.str == "T2"] <- 1
>   myWilcox <-
+   function(x, memSubjects, alpha = 0.05)
+   {
+     xc <- x[memSubjects == 1]
+     xn <- x[memSubjects == 0]
+
+     m <- sum(memSubjects == 1)
+     res <- wilcox.test(x = xc, y = xn, conf.level = 1 - alpha)
+     res2 <- c(res$p.value, res$statistic - m * (m + 1) / 2)
+     names(res2) <- c("p.value", "statistic")
+
+     return(res2)
+   }
>   tmp <- t(apply(mat, 1, myWilcox, memSubjects = memSubjects))
>   colnames(tmp) <- c("p.value", "statistic")
>   memIni <- rep(2, nrow(mat))
>   memIni[tmp[, 1] < 0.05 & tmp[, 2] > 0] <- 1
>   memIni[tmp[, 1] < 0.05 & tmp[, 2] < 0] <- 3
>   print(table(memIni))
>   obj.gsMMD <- gsMMD2.default(mat, memSubjects, memIni = memIni,
+           transformFlag = TRUE, transformMethod = "boxcox", scaleFlag = TRUE,
+           quiet=TRUE)
>   para <- obj.gsMMD$para
>   print(round(para, 3))
>
>
```

Actually, the two sample Wilcoxon rank-sum test is implemented in the *GeneSelectMMD*. The above code is used as an illustration on how to use the functions `gsMMD2` and `gsMMD2.default`.

Note that the speed of the four functions can be slow for large data sets, however it has been improved since version 2.0.7 by using Fotran code.

5 Error rates estimated for real microarray data sets

For real microarray data sets, the classical gene selection methods, such as two-sample t-test, two-sample Wilcoxon rank-sum test, do not directly provide the estimates of error rates such as false discovery rate

(denoted as FDR; it is the percentage of nondifferentially expressed genes among selected genes), false non-discovery rate (denoted as FNDR; it is the percentage of differentially expressed genes among unselected genes), false positive rate (denoted as FPR; it is the percentage of selected genes among nondifferentially expressed genes), and false negative rate (denoted as FNR; it is the percentage of un-selected genes among differentially expressed genes), since the true gene cluster membership is unknown.

However, model-based gene selection methods (e.g., eLNN and eGG proposed by Lo and Gottardo (2007), and the method proposed by Qiu et al.'s (2008)) could easily estimate these error rates, since these error rates are functions of some probabilities which are in turn the functions of model parameters.

The function `errRates` is used to estimate FDR, FNDR, FPR, and FNR based on the objects returned by the four gene selection functions mentioned in the previous section. This function returns a 4×1 vector with elements FDR, FNDR, FPR, and FNR.

For example,

```
>   print(round(errRates(obj.gsMMD), 3))

  FDR  FNDR   FPR   FNR
0.027 0.000 0.003 0.000

>
```

6 Visualizing the fit of the model to a real microarray data set

In general, it is difficult to visualize the fit of the model to a real microarray data set since the data are in high-dimensional space. However, it is possible to do so for the special case where the tissue samples within a tissue type are marginally independent. In this case, we can pool all gene expression levels together since they are all independent, and regard them as one-dimensional data. Next, we can impose the density estimate onto the histogram of this pooled data.

The function `plotHistDensity` is used for such purpose. The following R code illustrates the usage of `plotHistDensity`:

```
>   plotHistDensity(obj.gsMMD, plotFlag = "case",
+       mytitle = "Histogram of gene expression levels for T2\nimposed with estimated density (case)",
+       plotComponent = TRUE,
+       x.legend = c(0.8, 3),
+       y.legend = c(0.3, 0.4),
+       numPoints = 500,
+       cex.main = 1,
+       cex = 1)
>
```

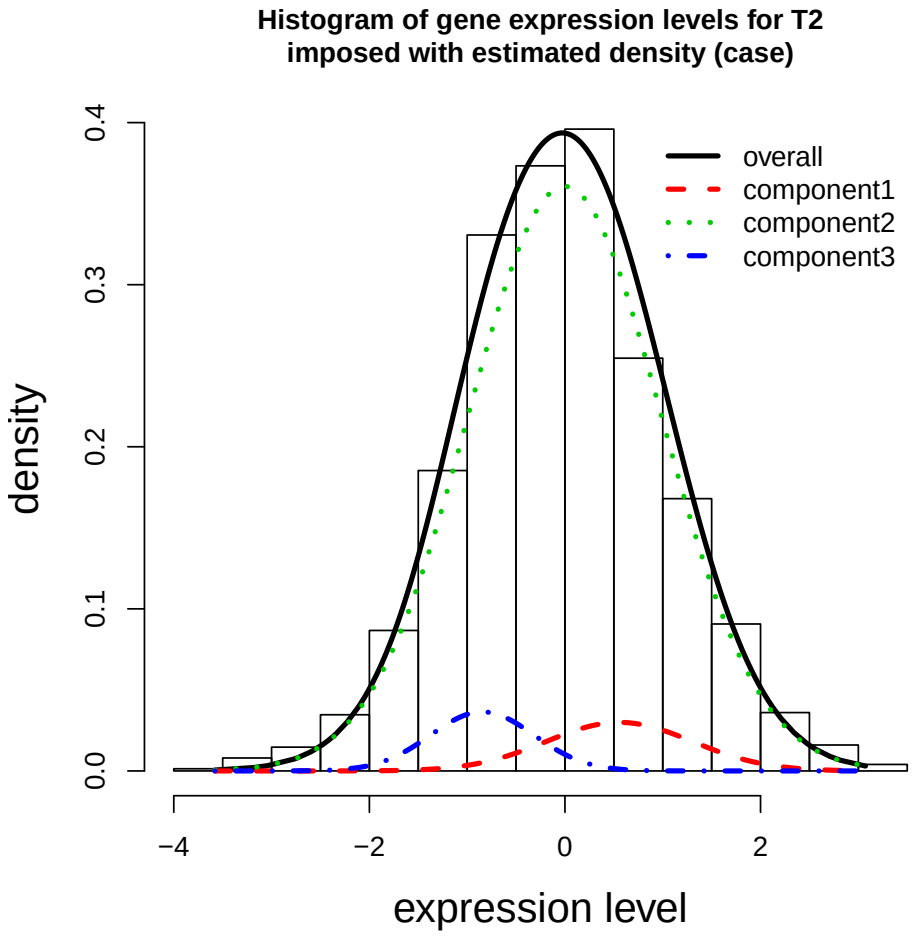

Figure 1: Plot produced using plotHistDensity

7 Efficiency improvements

Version 2.0.7 includes efficiency improvements to the EM optimization process. Specifically, Fortran 77 code for version 3.0 of the L-BFGS-B optimization algorithm from Morales and Nocedal (2011) has been included. The objective functions for the EM optimization have been translated to Fortran 77 in order to facilitate the use of this optimization code. Additional operations within the EM optimization iterative loop have also been written in Fortran 77 to improve the speed, where the calculation of the weigths has been addressed. Lastly, although the t-test is an optional operation, and therefore, improvements in its speed may not always be of consequence, the two-sample t-test calculations have increased efficiency in version 2.0.7.

Analysis involving simulated microarray data for 32,000 genes and a total of 2,000 samples is accomplished in under a minute on a notebook computer (Figure **??**), and therefore, the enhanced performance provided by GeneSelectMMD can be attained on larger datasets using typical hardware.

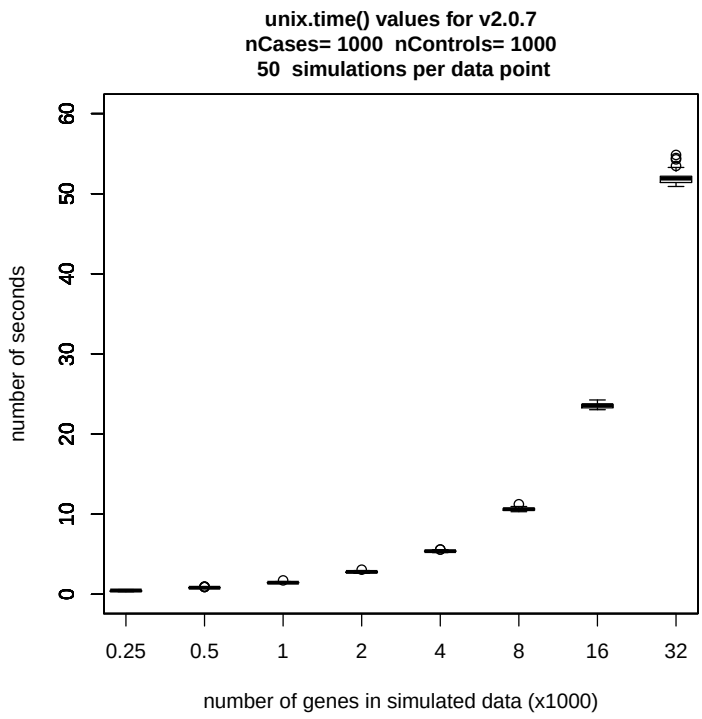

Figure 2: Plot of running time vs number of genes using an Asus notebook with Intel® Core™ i5-2450M CPU and 8GB memory

8 Discussion

The speeds of the four gene selection functions described in Section **??** can be slow, however the speed has been improved by embedding Fortran code in the R code for version 2.0.7 of the *GeneSelectMMD*.