# The Normal Uniform Differential Gene Expression (*nudge*) detection package

Nema Dean

17th March 2006

## Contents

## 1 Overview

The nudge package can be used to estimate the probabilities of genes being differentially expressed in two samples with one or more replicates. The probability of differential expression comes from modeling the normalized data with a two component mixture model; a uniform component for the differentially expressed genes and a normal component for the other genes (see Dean and Raftery (2005) for details). Posterior probabilities of being in the differentially expressed group are computed using the estimated model parameters. Typically a threshold for probability of being expressed of 0.5 is used for classification of the genes into the two groups ($\geq 0.5$ = differentially expressed, $< 0.5$ = not differentially expressed) but the probabilities can also be used to rank the genes in terms of how likely they are to be differentially expressed. We begin by loading the library

```
> library(nudge)
```

## 2 Single replicate data: the nudge1 function

The input for the nudge1 function is the single column matrices or vectors of log ratios and log intensities. The logarithm is to the base 2. An example of single replicate data is the like-like data (see Section 6.1 for details) but we must first manipulate the data into the correct form. The log ratio vector, lR, is the log (to the base 2) of the ratio of the two sets of intensity vectors, like[,1] & like[,2], or by the rules of logarithms, the difference between the log of each vector. The log intensity, lI, is the log (to the base 2) of the product of the two sets of intensity vectors, like[,1] & like[,2], or by the rules of logarithms, the sum of the log of each vector.

```
> data(like)
> lR <- log(like[, 1], 2) - log(like[, 2], 2)
> lI <- log(like[, 1], 2) + log(like[, 2], 2)
```

The nudge1 function will automatically detect that there is only a single replicate and adjust the normalization performed within the function accordingly. The default parameters for the normalization are recommended for use in general but can be changed if necessary. The *span1* parameter is the proportion of data to be used in the loess mean normalization (default is 0.6), the *span2* parameter is the proportion of data to be used in the loess variance normalization (default is 0.2). More details about the normalizations are available in Dean and Raftery (2005).

The tolerance for convergence of the log-likelihood of the fitted model in the EM algorithm can also be adjusted to allow faster or slower computation times (default is 0.00001), also a limiting number of iterations that the EM fitting algorithm is allowed to run for is specified, in case the convergence is slow and the algorithm needs to be stopped before convergence is reached (default is 500). In practice, the number of iterations to convergence is rarely more than 10 and the EM algorithm estimation part of the function takes much less time to run than the loess normalizations. Generally the method takes about 30 seconds to run. To get the fitted results for the data we run the following:

```
> result <- nudge1(logratio = lR, logintensity = lI)
```

The nudge1 function produces a list including the vector of probabilities of differential expression for all the genes, pdiff, the matrix of normalized log ratios, lRnorm, estimates of the parameters of the model, mu (the EM estimate of the mean of the normal non-differentially expressed group of normalized log ratios), sigma (the EM estimate of the standard deviation of the normal non-differentially expressed group of normalized log ratios), mixprob (the prior or mixing probability for a gene not being differentially expressed), a and b (the min and max respectively of the normalized log ratios, giving the range of the uniform mixture component), the converged log likelihood value for the fitted model, loglike, and the number of iterations that the EM algorithm ran for, iter.

```
> names(result)

[1] "pdiff"   "lRnorm"  "mu"      "sigma"   "mixprob" "a"       "b"
[8] "loglike" "iter"
```

If the genes are labeled we can look at the names or if not the number of the genes with the highest probability of differential expression. We do this by using the `sort` command as follows to look at the row number and probabilities of the genes with the 20 highest probabilities of differential expression.

```
> s <- sort(result$pdiff, decreasing = T, index.return = T)
> names(lR) <- c(1:length(lR))
> cbind(names(lR)[s$ix[1:20]], round(s$x[1:20], 2))
```

```
      [,1]   [,2]
 [1,] "4604" "1"
 [2,] "4397" "1"
 [3,] "4513" "1"
 [4,] "5603" "1"
 [5,] "5500" "1"
 [6,] "1557" "1"
 [7,] "1758" "1"
 [8,] "4539" "1"
 [9,] "1617" "1"
[10,] "3572" "1"
[11,] "1121" "1"
[12,] "3479" "0.99"
[13,] "3466" "0.99"
[14,] "4449" "0.98"
[15,] "7393" "0.98"
[16,] "3817" "0.96"
[17,] "669"  "0.95"
[18,] "5569" "0.92"
[19,] "5595" "0.88"
[20,] "5379" "0.87"
```

We can also look at the number of genes found to be significant in the sense that their probability of differential expression is greater than a given threshold (usual threshold, `thresh` used here is 0.5). We can also look at the row numbers or names of these genes.

```
> thresh <- 0.5
> sum(result$pdiff >= thresh)
```

```
[1] 29
```

```
> names(lR)[result$pdiff >= thresh]
```

```
 [1] "185"  "669"  "769"  "1121" "1557" "1617" "1758" "3266" "3466" "3479"
[11] "3572" "3817" "4397" "4449" "4513" "4539" "4604" "4731" "4801" "4807"
[21] "5062" "5379" "5500" "5569" "5595" "5603" "6725" "6726" "7393"
```

One can also look at the plot of the original log ratios, `lR`, versus the log intensities, `lI`, compared to the plot of the normalized log ratios, `lRnorm`, produced by the normalization method versus the log intensities, `lI`, (along with the loess fitted mean lines of both).

```
> lRnorm <- result$lRnorm
```

```
> par(mfrow = c(1, 2))
> yl <- range(lR, lRnorm)
> plot(lI, lR, pch = ".", main = "Log ratios versus log intensities",
+     xlab = "log intensities", ylab = "log ratios", ylim = yl)
> l <- loess(lR ~ lI, span = 0.6)
> slI <- sort(lI, index.return = T)
> lines(cbind(slI$x, l$fitted[slI$ix]))
> plot(lI, lRnorm, pch = ".", main = "Normalized log ratios versus log intensities",
+     xlab = "average log intensities", ylab = "normalized log ratios",
+     ylim = yl)
> l <- loess(lRnorm ~ lI, span = 0.6)
> slI <- sort(lI, index.return = T)
> lines(cbind(slI$x, l$fitted[slI$ix]))
```

These plots are given in Figure 1 at the end of this document.

# 3 Multiple replicate data: the `nudge1` function

The input for the `nudge1` function is the matrices of log ratios and log intensities. The logarithm is to the base 2. An example of multiple replicate data where the two samples are labeled with different dyes and then hybridized to arrays is the HIV dataset (see Section 6.2 for details). We must first manipulate the data into the correct form. `lR` is the matrix of log ratios to the base 2 which we get by subtracting the logs of the intensities for sample 1 for each gene from the logs of the corresponding intensities for sample 2 for each gene (by corresponding intensities we mean those paired to the same slide). `lI` is the matrix of log intensities to the base 2 which we get by adding the logs of the intensities for sample 1 for each gene from the logs of the corresponding intensities for sample 2 for each gene (by corresponding intensities we mean those paired to the same slide). Note that in the HIV dataset each of the first 4 columns is paired with each of the last 4 columns in order.

```
> data(hiv)
> lR <- log(hiv[, 1:4], 2) - log(hiv[, 5:8], 2)
> lI <- log(hiv[, 1:4], 2) + log(hiv[, 5:8], 2)
```

The function will automatically detect that there are multiple replicates in this dataset, giving multiple log ratios and intensities for each gene and adjust the normalization performed within the function accordingly.

The only necessary input apart from the log ratios or intensities is whether or not there is a balanced dye swap when there are multiple replicates. When we say a dye swap we mean that in one set of replicates sample 1 was labeled with red dye and sample 2 with green dye and in another set of replicates sample 2 was labeled with red dye and sample 1 with green dye. A balanced dye swap means that there are the same number of replicates in each of the two sets of dye-assignments. This is given by the *dye.swap* logical argument (default is FALSE, no balanced dye-swap).

If there is no dye-swap, a `loess` normalization of the mean will be done using *span1* proportion of the data. No mean normalization will be necessary if there is a balanced dye-swap. The argument *quant* is the quantity used to choose the constant that the (mean-normalized) average log ratio for a gene is divided by when the standard deviation across replicates for the log ratios of that gene is less than its absolute (mean-normalized) average log ratio (the constant is the $quant^{th}$ quantile of the distribution of standard deviations of the log ratios for all genes whose absolute (mean-normalized) average log ratio is greater than their standard deviation, the default is 0.99). If a gene's standard deviation for log ratios across replicates is greater than its absolute (mean-normalized) average log ratio then we simply divide the (mean-normalized) average log ratio by the gene's standard deviation. The default parameters for the normalization are recommended for use in general but can be changed if necessary.

The tolerance for convergence of the log-likelihood of the fitted model in the EM algorithm can also be adjusted to allow faster or slower computation times (default is 0.00001), also a limiting number of iterations that the EM fitting algorithm is allowed to run for is specified, in case the convergence is slow and the algorithm needs to be stopped before convergence is reached (default is 500). In practice, the number of iterations to convergence is rarely more than 10 and the EM algorithm estimation part of the function takes much less time to run than the `loess` normalizations. Generally the method takes about 30 seconds to run. In the case of the HIV data there is a balanced dye-swap so we would run the following:

```
> result <- nudge1(logratio = lR, logintensity = lI, dye.swap = T)
```

The `nudge1` function produces a list including the vector of probabilities of differential expression for all the genes, `pdiff`, the matrix of normalized log ratios, `lRnorm`, estimates of the parameters of the model, `mu` (the EM estimate of the mean of the normal non-differentially expressed group of normalized average log ratios), `sigma` (the EM estimate of the standard deviation of the normal non-differentially expressed group of normalized average log ratios), `mixprob` (the prior or mixing probability for a gene not being differentially expressed), `a` and `b` (the min and max respectively of the normalized average log ratios, giving the range of the uniform mixture component), the converged log likelihood value for the fitted model, `loglike`, and the number of iterations that the EM algorithm ran for, `iter`, the same as in the single replicate case.

```
> names(result)

[1] "pdiff"   "lRnorm"  "mu"      "sigma"   "mixprob" "a"       "b"
[8] "loglike" "iter"
```

If the genes are labeled again we can look at the names or if not the row numbers of the genes with the highest probability of differential expression, in the same way as in the single replicate case.

```
> s <- sort(result$pdiff, decreasing = T, index.return = T)
> rownames(lR) <- c(1:nrow(lR))
> cbind(rownames(lR)[s$ix[1:20]], round(s$x[1:20], 2))

      [,1]   [,2]
 [1,] "3"    "1"
 [2,] "5"    "1"
 [3,] "7"    "1"
 [4,] "771"  "1"
 [5,] "773"  "1"
 [6,] "775"  "1"
 [7,] "1539" "1"
 [8,] "1541" "1"
 [9,] "2307" "1"
[10,] "2309" "1"
[11,] "2311" "1"
[12,] "3859" "1"
[13,] "2348" "1"
[14,] "1543" "0.96"
[15,] "3186" "0.94"
[16,] "1943" "0.73"
[17,] "1500" "0.27"
[18,] "21"   "0.22"
[19,] "1940" "0.17"
[20,] "48"   "0.14"
```

We can also look at the number (and names) of genes found to be significant in the sense that their probability of differential expression is greater than a given threshold (usual threshold used is 0.5).

```
> thresh <- 0.5
> sum(result$pdiff >= thresh)

[1] 16

> rownames(lR)[result$pdiff >= thresh]

 [1] "3"    "5"    "7"    "771"  "773"  "775"  "1539" "1541" "1543" "1943"
[11] "2307" "2309" "2311" "2348" "3186" "3859"
```

One can also look at the plot of the original average log ratios, `lRbar`, versus the average log intensities, `lIbar`, compared to the plot of the normalized average log ratios, `lRnorm`, produced by the normalization method versus the average log intensities, `lIbar`, (along with the loess fitted mean lines of both).

```
> lRbar <- apply(lR, 1, mean)
> lIbar <- apply(lI, 1, mean)
> lRnorm <- result$lRnorm

> par(mfrow = c(1, 2))
> yl <- range(lRbar, lRnorm)

>  plot(lIbar,lRbar,pch=".",main="Average log ratios
+  versus
+ average log  intensities",xlab="average log intensities",
+  ylab="average log ratios", ylim=yl)
```

```
> l <- loess(lRbar ~ lIbar, span = 0.6)
> slI <- sort(lIbar, index.return = T)
> lines(cbind(slI$x, l$fitted[slI$ix]))

>  plot(lIbar,lRnorm,pch=".",main="Normalized average log ratios
+  versus
+ average log intensities",xlab="average log intensities",
+  ylab="normalized average log ratios", ylim=yl)

> l <- loess(lRnorm ~ lIbar, span = 0.6)
> slI <- sort(lIbar, index.return = T)
> lines(cbind(slI$x, l$fitted[slI$ix]))
```

These plots are given in Figure 2 at the end of this document.

We can also check to see if, using the threshold of 0.5 we have correctly identified the positive control genes as differentially expressed and the negative control genes as not differentially expressed. We create `diff`, a vector of indicator variables with 1 indicating a gene is identified as differentially expressed, 0 otherwise (according to a threshold of 0.5 in our model) and `ndiff`, a vector of indicator variables with 1 indicating a gene is identified as not differentially expressed, 0 otherwise (according to a threshold of 0.5 in our model). We then use the similar vectors, `pos.contr` a vector of indicators for true (known) differential expression (positive control genes) and `neg.contr` a vector of indicators for true (known) non-differential expression (negative control genes). Below, we calculate the number of positive control genes we correctly identified as differentially expressed, the percentage of positive control genes identified as differentially expressed, the number of negative control genes we correctly identified as non-differentially expressed and the percentage of negative controls we correctly identified as non-differentially expressed.

```
> diff <- round(result$pdiff)
> ndiff <- 1 - diff
> sum((diff == pos.contr) & pos.contr)

[1] 13

> sum((diff == pos.contr) & pos.contr)/sum(pos.contr) * 100

[1] 100

> sum((ndiff == neg.contr) & neg.contr)

[1] 29

> sum((ndiff == neg.contr) & neg.contr)/sum(neg.contr) * 100

[1] 100
```

We can also calculate the fitted density of the model (`d` for range of points `x`) and compare it to the density histogram of the normalized data to look at how good the fit is.

```
> x <- seq((result$a - 1), (result$b + 1), 1e-04)
> d <- (1 - result$mixprob) * dunif(x, result$a, result$b) + result$mixprob *
+     dnorm(x, result$mu, result$sigma)
> par(mfrow = c(1, 2))
```

```
>  hist(result$lRnorm,freq=F,main="Histogram of
+  average normalized log ratios", xlab="average normalized log ratios",breaks=25)
```

```
> lines(cbind(x, d), lty = 2)
```

```
>  hist(result$lRnorm,freq=F,main="Right-side of the histogram of
+  average normalized log ratios", xlab="average normalized log ratios",
+  xlim=c(1.5, result$b),ylim=c(0,0.1),breaks=25)
```

```
> lines(cbind(x, d), lty = 2)
```

These plots are given in Figure 3 at the end of this document.

# 4 Multiple replicate data with samples compared to a reference sample: the nudge2 function

In this case we are comparing both samples with a reference sample. Let's call the two types of samples control and treatment. If the control replicates are samples that are from different biological organisms from the samples used in the treatment replicates, i.e. biological replicates, there is no sensible way to decide which control samples to compare in a ratio with which treatment samples. Instead, we compare all control and treatment samples to a reference sample and instead of looking at log ratios we look instead at differences between the average across-treatment replicates' log ratios and the average across-control replicates' log ratios to check for differential expression. In the Apo AI dataset the pooled control mRNA is used as a reference sample for both treatment (knockout) and control biological replicates (see Section 6.3 for details). The input for the `nudge2` function is the control and treatment log ratios and log intensities. We must first manipulate the data into the correct form.

```
>  apo<-read.csv(file=
+ "http://www.stat.berkeley.edu/users/terry/zarray/Data/ApoA1/rg_a1ko_morph.txt",
+  header=T)
```

```
> rownames(apo) <- apo[, 1]
> apo <- apo[, -1]
> colnames(apo)
```

```
 [1] "c1G" "c1R" "c2G" "c2R" "c3G" "c3R" "c4G" "c4R" "c5G" "c5R" "c6G" "c6R"
[13] "c7G" "c7R" "c8G" "c8R" "k1G" "k1R" "k2G" "k2R" "k3G" "k3R" "k4G" "k4R"
[25] "k5G" "k5R" "k6G" "k6R" "k7G" "k7R" "k8G" "k8R"
```

There are zero entries in this dataset so in order to be able to take logs we add 1 to all entries. We create: `lRctl` the matrix of control log ratios to the base 2, `lRtxt` the matrix of treatment log ratios to the base 2, `lIctl` the matrix of control log intensities to the base 2 and `lItxt` the matrix of treatment log intensities to the base 2.

```
> apo <- apo + 1
> lRctl <- log(apo[, c(seq(2, 16, 2))], 2) - log(apo[, c(seq(1,
+     15, 2))], 2)
> lRtxt <- log(apo[, c(seq(18, 32, 2))], 2) - log(apo[, c(seq(17,
+     31, 2))], 2)
> lIctl <- log(apo[, c(seq(2, 16, 2))], 2) + log(apo[, c(seq(1,
+     15, 2))], 2)
> lItxt <- log(apo[, c(seq(18, 32, 2))], 2) + log(apo[, c(seq(17,
+     31, 2))], 2)
```

The normalization of the mean will be done using *span1* proportion of the data (default is 0.6). The *quant* argument is the quantity used to choose the constant that the (mean-normalized) average log ratio differences for a gene is divided by when the standard deviation across replicates for the log ratio differences of that gene is less than its absolute (mean-normalized) average log ratio difference (the constant is the $quant^{th}$ quantile of the distribution of standard deviations of the log ratio differences for all genes whose absolute (mean-normalized) average log ratio difference is greater than their standard deviation, the default is 0.99). If a gene's standard deviation for log ratio differences across replicates is greater than its absolute (mean-normalized) average log ratio difference then we simply divide the (mean-normalized) average log ratio difference by the gene's standard deviation. The default parameters for the normalization are recommended for use in general but can be changed if necessary.

The tolerance for convergence of the log-likelihood of the fitted model in the EM algorithm can also be adjusted to allow faster or slower computation times (default is 0.00001), also a limiting number of iterations that the EM fitting algorithm is allowed to run for is specified, in case the convergence is slow and the algorithm needs to be stopped before convergence is reached (default is 500). In practice, the number of iterations to convergence is rarely more than 10 and the EM algorithm estimation part of the function takes much less time to run than the `loess` normalizations. Generally the method takes about 30 seconds to run. In the case of the Apo AI data we run the following:

```
> result <- nudge2(control.logratio = lRctl, txt.logratio = lRtxt,
+     control.logintensity = lIctl, txt.logintensity = lItxt)
```

As in the previous cases, the `nudge2` function produces a list including the vector of probabilities of differential expression for all the genes, `pdiff`, the matrix of normalized average log ratio differences, `lRnorm`, estimates of the parameters of the model, `mu` (the EM estimate of the mean of the normal non-differentially expressed group of normalized average log ratio differences), `sigma` (the EM estimate of the standard deviation of the normal non-differentially expressed group of normalized average log ratio differences), `mixprob` (the prior or mixing probability for a gene not being differentially expressed), `a` and `b` (the min and max respectively of the normalized average log ratio differences, giving the range of the uniform mixture component), the converged log likelihood value for the fitted model, `loglike`, and the number of iterations that the EM algorithm ran for, `iter`.

```
> names(result)

[1] "pdiff"   "lRnorm"  "mu"      "sigma"   "mixprob" "a"       "b"
[8] "loglike" "iter"
```

If the genes are labeled we can look at the names or if not the row numbers of the genes with the highest probability of differential expression.

```
> s <- sort(result$pdiff, decreasing = T, index.return = T)
> rownames(lRtxt) <- rownames(lRctl) <- c(1:nrow(lRtxt))
> cbind(rownames(lRtxt)[s$ix[1:20]], round(s$x[1:20], 2))
```

```
      [,1]   [,2]  
 [1,] "540"  "1"   
 [2,] "2149" "1"   
 [3,] "5356" "1"   
 [4,] "4139" "1"   
 [5,] "1739" "1"   
 [6,] "2537" "1"   
 [7,] "4941" "0.99"
 [8,] "1496" "0.83"
 [9,] "5986" "0.32"
[10,] "541"  "0.26"
[11,] "716"  "0.1" 
[12,] "2538" "0.09"
[13,] "1224" "0.06"
[14,] "799"  "0.06"
[15,] "1204" "0.06"
[16,] "3729" "0.05"
[17,] "1337" "0.04"
[18,] "1"    "0.04"
[19,] "800"  "0.03"
[20,] "1222" "0.03"
```

We can also look at the number (and names) of genes found to be significant in the sense that their probability of differential expression is greater than a given threshold (usual threshold used is 0.5).

```
> thresh <- 0.5
> sum(result$pdiff >= thresh)
```

```
[1] 8
```

```
> rownames(lRtxt)[result$pdiff >= thresh]
```

```
[1] "540"  "1496" "1739" "2149" "2537" "4139" "4941" "5356"
```

One can also look at the plot of the original average log ratio differences, `lRbar`, versus the average log intensities, `lIbar`, compared to the plot of the normalized average log ratio differences, `lRnorm`, produced by the normalization method versus the average log intensities, `lIbar`, (along with the loess fitted mean lines of both).

```
> lRbar <- apply(lRtxt, 1, mean) - apply(lRctl, 1, mean)
> lIbar <- apply(cbind(lItxt, lIctl), 1, mean)
> lRnorm <- result$lRnorm
> par(mfrow = c(1, 2))
> yl <- range(lRbar, lRnorm)

> plot(lIbar,lRbar,pch=".",main="Average log ratio differences
+  versus
+ average log intensities",xlab="average log intensities",
+  ylab="average log ratio differences", ylim=yl)

> l <- loess(lRbar ~ lIbar, span = 0.6)
> slI <- sort(lIbar, index.return = T)
> lines(cbind(slI$x, l$fitted[slI$ix]))

> plot(lIbar,lRnorm,pch=".",main="Normalized average log ratio differences
+  versus
+ average log intensities",xlab="average log intensities",
+  ylab="normalized average log ratio differences",ylim=yl)

> l <- loess(lRnorm ~ lIbar, span = 0.6)
> slI <- sort(lIbar, index.return = T)
> lines(cbind(slI$x, l$fitted[slI$ix]))
```

These plots are given in Figure 4 at the end of this document.

# 5 Including starting estimate information in the nudge functions

It has been shown that the EM algorithm (used to fit this model) can be sensitive to the starting values in terms of the estimates it converges to. The default starting estimates used in this package are that any gene, whose absolute (mean and spread) normalized log ratio (or log ratio difference) is greater than the mean of the (mean and spread) normalized log ratios for all genes plus two standard deviations of the (mean and spread) normalized log ratios across all genes, has its starting label for the differentially expressed group set to one and the non differentially expressed group set to zero. All other genes have their starting label for the differentially expressed group set to zero and the non differentially expressed group set to one. We may have some better information about the data we are analysing than this simple default, which can be translated into better starting values for the group labels.

For example we may believe that in the like-like data we have only 10 (say) differentially expressed genes. We may not know which genes those are but we have an idea of the prior probability of being differentially expressed ($\frac{10}{7680}$). We can include this information in the starting group label estimates for the EM algorithm by randomly generating the starting estimated value of the group labels matrix in the following way:

```
> data(like)
> lR <- log(like[, 1], 2) - log(like[, 2], 2)
> lI <- log(like[, 1], 2) + log(like[, 2], 2)
> p <- 10/nrow(like)
> temp <- rbinom(nrow(like), 1, p)
> z <- matrix(c(1 - temp, temp), nrow(like), 2, byrow = F)
> result <- nudge1(logratio = lR, logintensity = lI, z = z)
```

It is important that the labels for the differentially expressed group be in the second column of z and the labels for the not differentially expressed group be in the first, otherwise you will be giving prior information that is the opposite of what you intend. Note that because the mixing parameters are estimated from the counts of this matrix you are essentially specifying this. Because the EM can converge to local maxima, if you are using a random start like this, it is recommended that you generate several different z starting matrices and run the algorithm several times, selecting the result that gives the highest log likelihood value. Note also, that these group labels are only starting values and will be re-estimated iteratively by the EM algorithm, so the same values will not necessarily be returned.

Suppose you have even more specific information. Namely that you know (or strongly believe that certain genes are differentially expressed), this can also be incorporated into the algorithm. Say in the HIV dataset that we stronly believe that the genes in rows 3, 5, 7, 771, 773, 775, 1539, 1541, 1543, 2307, 2309, 2311 and 3859 are differentially expressed (note that these are the positive controls) and that we believe that the genes in rows 2348, 3186 and 1943 are differentially expressed and that we strongly believe that genes in rows 3075, 3077, 3079, 3843 and 3845 are not differentially expressed (note that these are some of the negative control genes) and that we suspect that most of the rest are not differentially expressed. We can create a starting group label matrix to incorporate this in the following way:

```
> data(hiv)
> lR <- log(hiv[, 1:4], 2) - log(hiv[, 5:8], 2)
> lI <- log(hiv[, 1:4], 2) + log(hiv[, 5:8], 2)
> p <- 1e-04
> temp <- rbinom(nrow(hiv), 1, p)
> z <- matrix(c(1 - temp, temp), nrow(hiv), 2, byrow = F)
> z[c(3, 5, 7, 771, 773, 775, 1539, 1541, 1543, 2307, 2309, 2311,
+     3859), 1] <- 0
> z[c(3, 5, 7, 771, 773, 775, 1539, 1541, 1543, 2307, 2309, 2311,
+     3859), 2] <- 1
> z[c(2348, 3186, 1943), 1] <- 0.05
> z[c(2348, 3186, 1943), 2] <- 0.95
> z[c(3075, 3077, 3079, 3843, 3845), 1] <- 1
> z[c(3075, 3077, 3079, 3843, 3845), 2] <- 0
> result <- nudge1(logratio = lR, logintensity = lI, z = z)
```

Again, generating several such matrices and re-running the `nudge1` function and selecting the result that gives the highest log likelihood may be advisable. It is again important to note that the generated labels you give the algorithm will not remain fixed but will be iteratively re-estimated like the other parameters in the EM algorith, so the same values will not necessarily be returned.

The same option is available in the `nudge2` function.

# 6 Data

## 6.1 Like-like dataset

This dataset is from a microarray experiment where the same samples (with different dyes) were hybridized to an array with 7680 genes. The expression levels in the red and green dyes were extracted from the image using customized software written at the University of Washington (Spot-On Image, developed by R. E. Bumgarner and Erick Hammersmark). The genes should be equally highly expressed, as each sample is the same, so ideally we should find few differentially expressed genes.

```
> rm(list = ls())
> data(like)
> ls()

[1] "like"

> dim(like)

[1] 7680    2
```

## 6.2 HIV dataset

This data consists of cDNA from CD4+ T cell lines at 1 hour after infection with HIV-1BRU; see van't Wout et al. (2003) for details. It is useful in testing the specificity and sensitivity of methods for identifying differentially expressed genes, since there are 13 genes known to be differentially expressed (HIV-1 genes), called positive controls, and 29 genes known not to be (non-human genes), called negative controls. There are 4608 gene expression levels recorded in each replicate. There are four replicates with balanced dye swaps (see the third paragraph in Section 3 for an explanation of balanced dye swaps) . We load the data.

```
> rm(list = ls())
> data(hiv)
> ls()

[1] "hiv"       "neg.contr" "pos.contr"

> dim(hiv)

[1] 4608    8

> length(pos.contr)

[1] 4608

> length(neg.contr)

[1] 4608
```

The first two columns of hiv are the expression measurements of Sample 1 under the first dye labeling scheme, the third and fourth columns are the expression measurements of Sample 1 under the second dye labeling scheme, the fifth and sixth columns are the expression measurements of Sample 2 under the first dye labeling scheme and the last two columns are the expression measurements of Sample 2 under the second labeling scheme. pos.contr is an indicator vector with entries set to 1 if the corresponding gene row in hiv belongs to a positive control and 0 otherwise. neg.contr is an indicator vector with entries set to 1 if the corresponding gene row in hiv belongs to a negative control and 0 otherwise.

## 6.3 Apo AI Dataset

This dataset was analyzed in Dudoit et al. (2002) and 8 genes were suggested to be differentially expressed. The data was obtained from 8 mice with the Apo AI gene knocked out and 8 normal mice. However the replicates were not created simply by comparing samples from control labeled with one dye versus knock-out mice labeled with the other. Instead, cDNA was created from samples from each of the 16 mice (both control and knock-out) and labeled in each case with the red dye. The green dye was used in all cases on cDNA created by pooling mRNA from all 8 control mice. Here the quantity of interest is not the log ratio of samples but the difference between the group averaged samples' log ratios. This data can be accessed from Terry Speed's website in the following way:

```
> rm(list = ls())
```

```
>  apo<-read.csv(file=
+  "http://www.stat.berkeley.edu/users/terry/zarray/Data/ApoA1/rg_a1ko_morph.txt",
+  header=T)
```

```
> dim(apo)
```

```
[1] 6384   33
```

Note that, because there are some zeros in the data, we must add a small positive constant in order to be able to take logs. We choose to add 1.

```
> apo[, -1] <- apo[, -1] + 1
```

# References

Dean, N. and A. E. Raftery (2005). Normal uniform mixture differential gene expression detection in cDNA microarrays. *BMC Bioinformatics 6*, 173–186.

Dudoit, S., Y. H. Yang, M. Callow, and T. Speed (2002). Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments. *Stat. Sin. 12*, 111–139.

van't Wout, A. B., G. K. Lehrma, S. A. Mikheeva, G. C. O'Keeffe, M. G. Katze, R. E. Bumgarner, G. K. Geiss, and J. I. Mullins (2003). Cellular gene expression upon human immunodeficiency virus type 1 infection of CD4+–T–Cell lines. *J. Virol. 77*, 1392–1402.

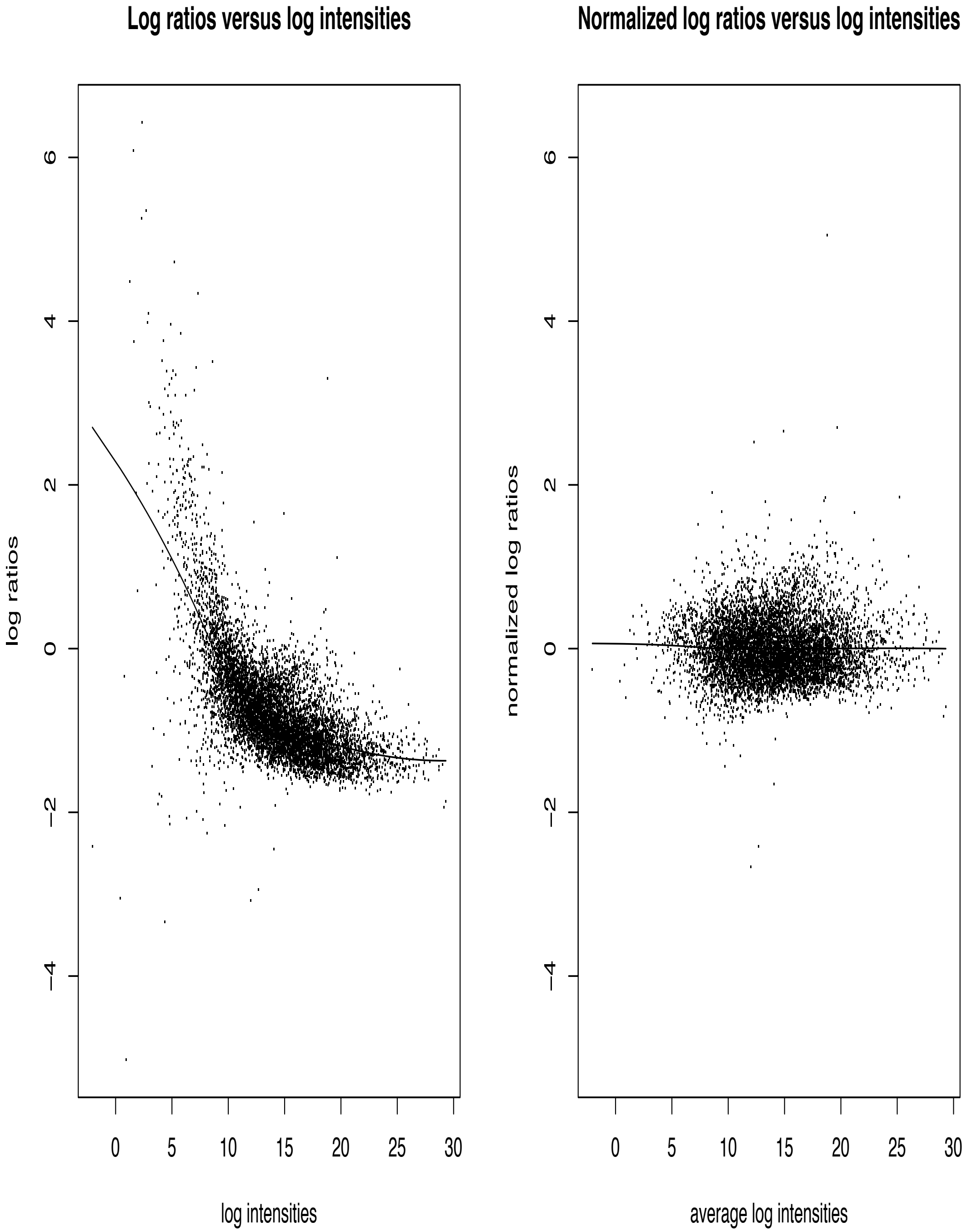

Figure 1: Plots of the original and normalized log ratios versus the log intensities for the like-like data

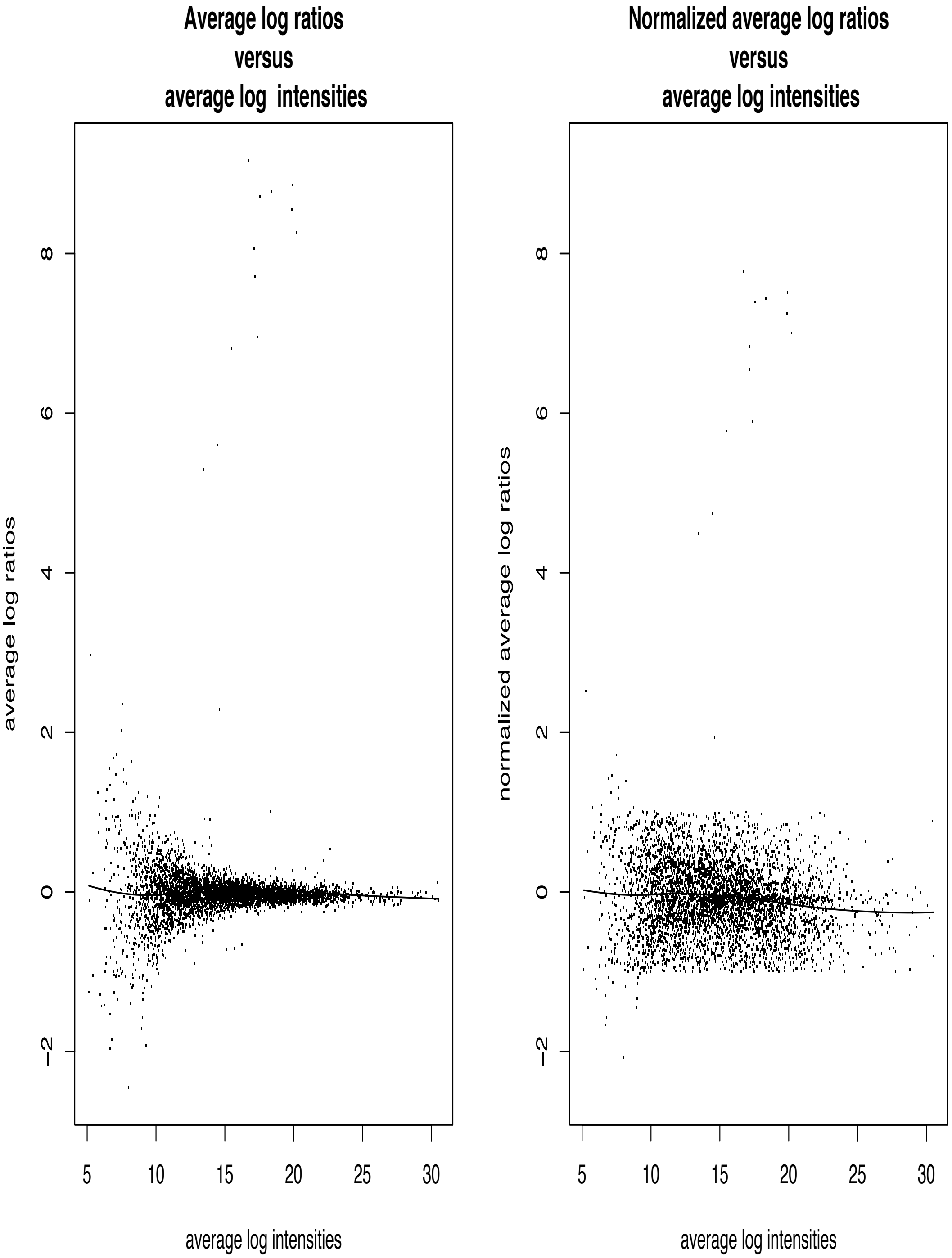

Figure 2: Plots of the original and normalized average log ratios versus the average log intensities for the HIV data

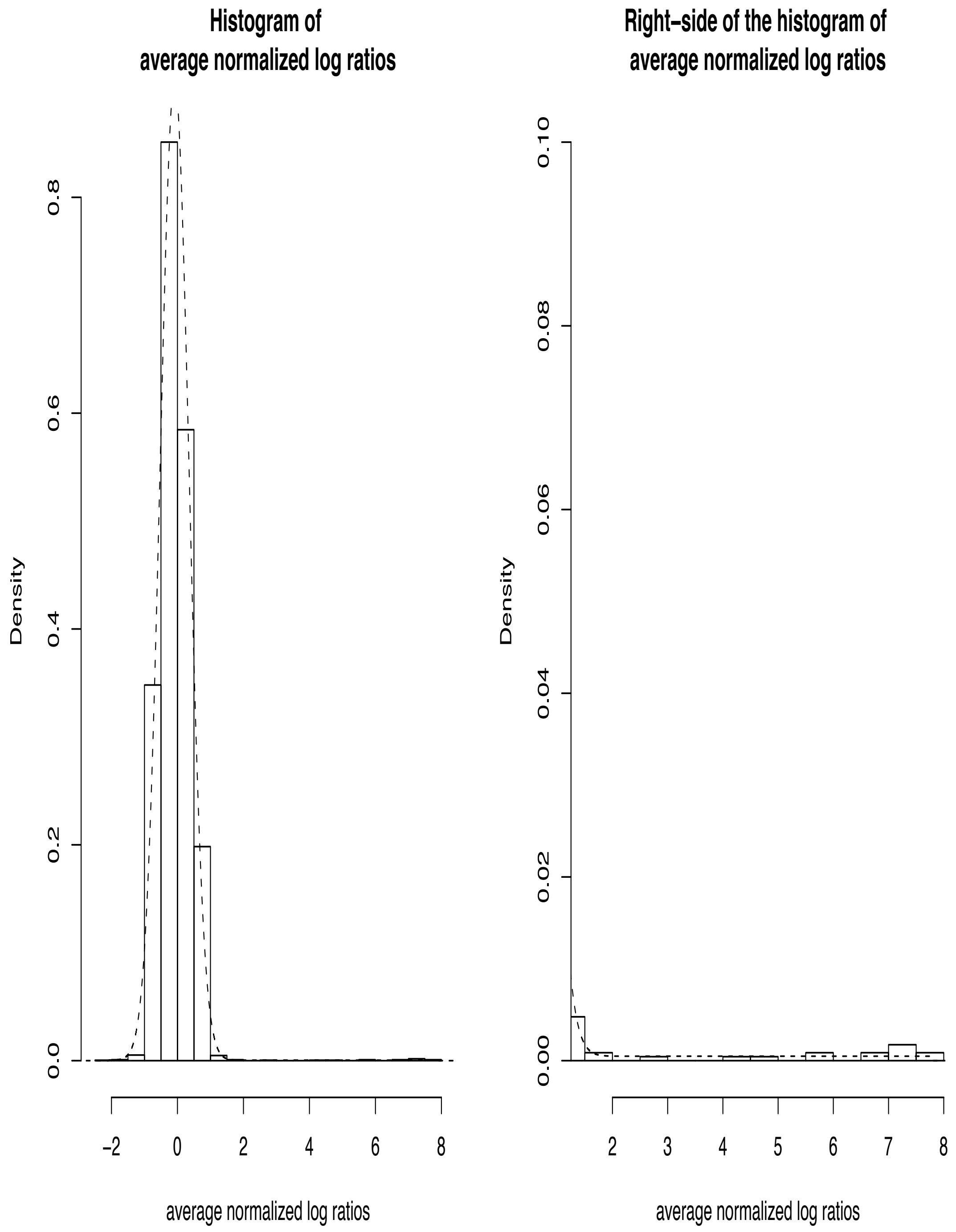

Figure 3: Histograms of the normalized average log ratios versus fitted model's density (given by the dashed line)

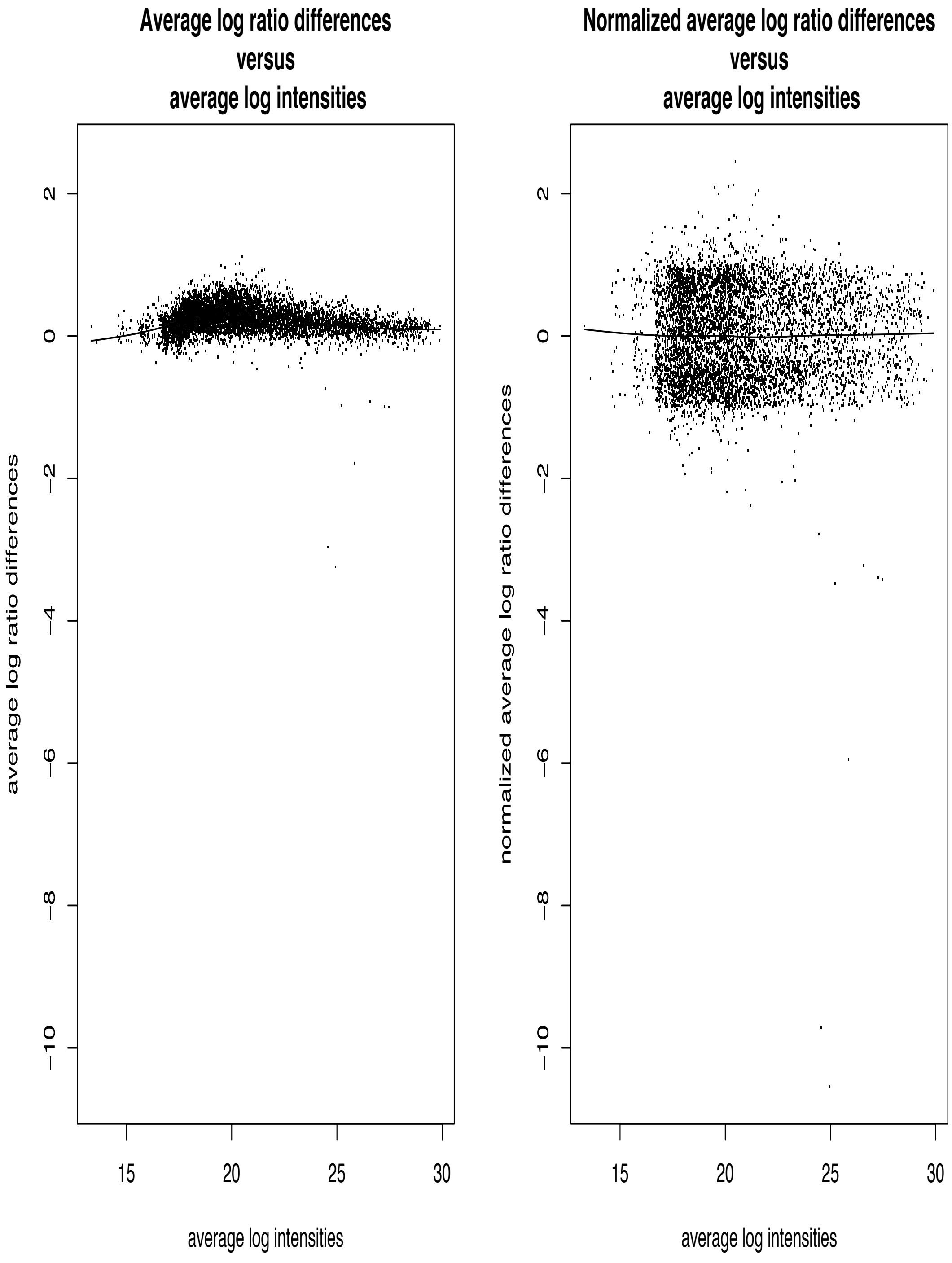

Figure 4: Plots of the original and normalized average log ratio differences versus the average log intensities for the Apo AI data