

Identifying differentially expressed genes with **siggenes**

Holger Schwender
holger.schw@gmx.de

Abstract

In this vignette, we show how the functions contained in the **R** package **siggenes** can be used to perform both the Significance Analysis of Microarrays (SAM) proposed by Tusher et al. (2001) and the Empirical Bayes Analysis of Microarrays (EBAM) suggested by Efron et al. (2001).

1 Introduction

Both the Significance Analysis of Microarrays (SAM) proposed by Tusher et al. (2001) and the Empirical Bayes Analysis of Microarrays (EBAM) suggested by Efron et al. (2001) can be used to identify differentially expressed genes and to estimate the False Discovery Rate (FDR). They are, however, not restricted to gene expression data, but can also be applied in any other multiple testing situation. Since SAM and EBAM have been developed for high-dimensional data, it, however, might be better to use more classical multiple testing approaches such as Bonferroni correction if there are only a few variables that should be tested.

In this vignette, it is described how the functions implemented in the **R** package **siggenes** can be used to perform a SAM or an EBAM analysis. For details on the algorithms behind these functions, see Schwender (2003), Schwender et al. (2003), and Schwender (2007).

As usual, it is necessary to load the package.

```
> library(siggenes)
```

In the following, we use the Golub et al. (1999) data set as it is provided by the `multtest` package to illustrate how the SAM and the EBAM analyses can be performed.

```
> data(golub)
```

`data(golub)` consists of a $3,051 \times 38$ matrix `golub` containing the expression levels of 3,051 genes and 38 samples, a vector `golub.cl` containing the class labels of the 38 samples, and a $3,051 \times 3$ matrix `golub.gnames` whose third column consists of the names of the genes.

Another example of how SAM can be applied to an `ExpressionSet` object can be found in Schwender et al. (2006). This article is part of a special issue of RNews that can be obtained by available

```
> browseURL("http://cran.r-project.org/doc/Rnews/Rnews_2006-5.pdf")
```

This article is also available in the doc-folder of the `siggenes` package and can be opened by

```
> openPDF(file.path(find.package("siggenes"), "doc", "siggenesRnews.pdf"))
```

2 General Information

While a SAM analysis starts by calling the function `sam`, an EBAM analysis starts – depending on whether an optimal choice for the fudge factor a_0 should be specified or not – either with `find.a0` or `ebam`. This is due to the fact that in SAM the fudge factor – if required – is computed during the actual analysis, whereas in EBAM it has to be determined prior to the actual analysis.

In `siggenes`, functions for performing a SAM or an EBAM analysis with moderated t -statistics (one-class or two-class, paired or unpaired, assuming equal or unequal group-variances), a moderated F -statistic, Wilcoxon rank statistics (one-class or two-class, paired or unpaired), and Pearson's χ^2 -statistic as well as the Cochran-Armitage trend test statistic C (for analyzing categorical data such as SNP data) are available. `siggenes` also provides the possibility to write your own function for using your own test score in `sam`, `find.a0` and `ebam` (see Chapter 6 for how to write such functions).

The different test scores can be selected by specifying the argument `method` of `sam` and `ebam`. In Table 1, the specifications of `method` for the different analyses are summarized.

Table 1: Specification of the argument `method` for the different analyses.

Test Score	<code>sam</code>	<code>ebam</code>	<code>find.a0</code>
t	<code>d.stat</code>	<code>z.ebam</code>	<code>z.find</code>
F	<code>d.stat</code>	<code>z.ebam</code>	<code>z.find</code>
Wilcoxon	<code>wilc.stat</code>	<code>wilc.ebam</code>	–
χ^2	<code>chisq.stat</code>	<code>chisq.ebam</code>	–
C	<code>trend.stat</code>	<code>trend.ebam</code>	–

Note that for `find.a0` only a function for the moderated t - and F -scores is available, since the fudge factor is not required – neither in EBAM nor in SAM – if Wilcoxon rank sums or Pearson’s χ^2 -statistic are employed.

To get the general arguments of `sam`, `ebam` and `find.a0`, `args` or `?` can be used, whereas method-specific arguments can be obtained by `args(foo)` or `?foo`, where `foo` is one of the functions/methods summarized in Table 1. So, e.g.,

```
> args(sam)
```

```
function (data, cl, method = d.stat, control = samControl(),
  gene.names = dimnames(data)[[1]], ...)
NULL
```

gives you the general arguments for a SAM analysis, whereas

```
> args(d.stat)
```

```
function (data, cl, var.equal = FALSE, B = 100, med = FALSE,
  s0 = NA, s.alpha = seq(0, 1, 0.05), include.zero = TRUE,
  n.subset = 10, mat.samp = NULL, B.more = 0.1, B.max = 30000,
  gene.names = NULL, R.fold = 1, use.dm = TRUE, R.unlog = TRUE,
  na.replace = TRUE, na.method = "mean", rand = NA)
NULL
```

returns the additional arguments for a SAM analysis with a moderated t - or F -statistic. These arguments can also be specified in `sam` because of the `...` in `sam`.

Note that one of the arguments of `sam` (and of `ebam` and `find.a0`) is `control`, which is essentially a list of further arguments for controlling the respective analysis. These further arguments can typically be ignored, as they are only provided for very special purposes (e.g., for trying to reproduce the results of an application of the Excel SAM version), and their defaults typically work well. All these arguments should only be changed if their meaning is completely understood, as their defaults have been chosen sensibly.

For `d.stat`, the most important arguments are `data` and `c1` (see Section 3). Additionally, `B`, `var.equal`, and `rand` might also be of importance. All the other arguments should only be changed if their meaning is completely understood, as their defaults have been chosen sensibly.

To obtain, e.g., the general help file for an EBAM analysis, use

```
> ?ebam
```

and to get the `z.ebam`-specific help files, use

```
> ?z.ebam
```

The arguments or the help files for the class-specific methods `plot`, `print`, `summary`, ... can be received by calling the functions `args.sam`, `args.ebam`, and `args.finda0`, or `help.sam`, `help.ebam`, and `help.finda0`. So, e.g.,

```
> args.sam(summary)
```

```
function(object, delta = NULL, n.digits = 5, what = "both", entrez = FALSE,
  bonf = FALSE, chip = "", file = "", sep = "\t", quote = FALSE, dec = ".")
```

returns all arguments of the `SAM`-class specific method `summary`, and

```
> help.finda0(plot)
```

opens an html-file containing the help for the `FindA0`-class specific method `plot`.

In the analysis with `sam` and `ebam`, all required statistics for a SAM or an EBAM analysis, respectively, are computed. Additionally, the number of differentially

expressed genes and the estimated FDR is determined for an initial (set of) value(s) for the threshold `delta`.

Note that even though the threshold for calling a gene differentially expressed is called `delta` in both `sam` and `ebam`, its meaning is totally different in both analyses: In SAM, `delta` is the distance between the observed and the expected (ordered) test scores, whereas in EBAM `delta` is the (posterior) probability that a gene with a specific test score is differentially expressed.

After the analysis with `sam` and `ebam`, the following functions can be employed for obtaining general and gene-specific information:

- `print`: Obtain the number of identified genes and the estimated FDR for other values of `delta`.
- `summary`: Get specific information on the genes called differentially expressed for a specified value of `delta`.
- `plot`: Generate a SAM plot or a plot of the posterior probabilities.
- `findDelta`: For a given value of the FDR, find the value of `delta` – and thus the number of identified variables – that provides the control of the FDR at the specified level. Alternatively, find the value of `delta` – and thus the level at which the FDR is controlled – for a specified number of variables.
- `sam2excel/ebam2excel`: Generate a csv-file containing the information returned by `summary` for the use in Excel.
- `sam2html/ebam2html`: Generate an html-file containing the information on the genes called differentially expressed with links to public repositories.
- `list.siggenes`: Get the names (as a character vector) of the genes called differentially expressed when using a specific value of `delta`.
- `link.siggenes`: Generate an html-file containing the output of `list.siggenes` with links to public repositories.

More details on these functions are given in the Chapters 4 and 5.

3 Required Arguments: `data` and `c1`

In the first step of both a SAM and an EBAM analysis, two arguments are required: `data` and `c1`.

The first required argument, `data`, is the matrix (or the data frame) containing the gene expression data that should be analyzed. Each row of this matrix must correspond to a gene, and each column must correspond to a sample. `data` can also be an `ExpressionSet` object (e.g., the output of `rma` or `gcrma`).

The second required argument, `c1`, is the vector of length `ncol(data)` containing the class labels of the samples. In an analysis of two class paired data, `c1` can also be a matrix. If `data` is an `ExpressionSet` object, `c1` can also be the name of the column of `pData(data)` containing the class labels.

The correct specification of the class labels depends on the type of data that should be analyzed. On the basis of this specification, the functions identify the type of data automatically.

One class data. In the one class case, `c1` is expected to be a vector of length n containing only 1's, where n denotes the number of samples but another value than 1 is also accepted. In the latter case, this value is automatically set to 1. So for $n = 10$, the vector `c1` is given by

```
> n <- 10
> rep(1, n)

[1] 1 1 1 1 1 1 1 1 1 1
```

Two class, unpaired data. In this case, the functions expect a vector `c1` consisting only of 0's and 1's, where all the samples with class label '0' belong to one group (e.g., the control group), and the samples with class label '1' belong to the other group (e.g., the case group). So if, for example, the first $n1 = 5$ columns of the data matrix correspond to controls and the next $n2 = 5$ columns correspond to cases, then the vector `c1` is given by

```
> n1 <- n2 <- 5
> rep(c(0, 1), c(n1, n2))

[1] 0 0 0 0 0 1 1 1 1 1
```

The functions also accept other values than 0 and 1. In this case, the smaller value is automatically set to 0, and the larger value to 1. So if, e.g., 1 is used as the label for group 1, and 2 for the label of group 2, then the functions will automatically set the class label '1' to 0, and the class label '2' to 1.

Two class, paired data. Denoting the number of samples by n , we here have $K = n/2$ paired observations. Each of the K samples belonging to the first group (e.g., the after treatment group) is labelled by one of the integers between 1 and K , and each of the K samples belonging to the other group (e.g., the before treatment group) is labelled by one of the integers between -1 and $-K$, where the sample with class label ' k ' and the sample with label ' $-k$ ' build an observation pair, $k = 1, \dots, K$. So if, e.g., the first $K = 5$ columns of the data matrix contain samples from the before treatment group, and the next $K = 5$ columns contain samples from the after treatment group, where the samples 1 and 6, 2 and 7, ..., respectively, build a pair, then the vector `c1` is given by

```
> K <- 5
> c((-1:-5), 1:5)

[1] -1 -2 -3 -4 -5  1  2  3  4  5
```

Another example: If the first column contains the before treatment measurements of an observation, the second column the after treatment measurements of the same observation, the third column the before treatment measurements of the second observation, the fourth column the after treatment measurements of the second observation, and so on, then a possible way to generate the vector `c1` for $K = 5$ paired observations is

```
> K <- 5
> rep(1:K, e = 2) * rep(c(-1, 1), K)

[1] -1  1 -2  2 -3  3 -4  4 -5  5
```

There is another way to specify the class labels in the two class paired case: They can be specified by a matrix with n rows and two columns. One of the column should contain -1 's and 1 's specifying the before and after treatment samples. The other column should consist of the integers between 1 and K indicating the observation pairs. So if we consider the previous example, `c1` can also be specified by

```
> K <- 5
> cbind(rep(c(-1, 1), 5), rep(1:5, e = 2))
```

	[,1]	[,2]
[1,]	-1	1
[2,]	1	1
[3,]	-1	2
[4,]	1	2
[5,]	-1	3
[6,]	1	3
[7,]	-1	4
[8,]	1	4
[9,]	-1	5
[10,]	1	5

While `c1` must be specify as described above if `c1` is a vector, other values will be accepted if `c1` is a matrix. In the latter case, the smaller value of the column of `c1` containing two different values will be set to -1, and the larger value to 1. The K different values in the other column are sorted and set to the integers between 1 and K .

Multi-class and categorical cases. In these cases, `c1` should be a vector containing the integers between 1 and g , where g is the number of different classes/levels. Other labels are also accepted, but will automatically be set to the integers between 1 and g .

4 Significance Analysis of Microarrays

In this section, we show how the Significance Analysis of Microarrays (SAM) proposed by Tusher et al. (2001) can be applied to the data set of Golub et al. (1999). Another example of how SAM can be applied to gene expression data can be found in Schwender et al. (2006). This article can be opened by calling

```
> openPDF(file.path(find.package("siggenes"),"doc","siggenesRnews.pdf"))
```

The matrix `golub` contains the expression values of 3,051 genes and 38 samples, while the vector `golub.c1` consists of the class labels that are either 0 and 1. Additionally, the gene names are provided by the third column of `golub.gnames`. (Note that if the row names of the data matrix already comprise the gene names, `gene.names` need not to be specified in `sam`.)

A SAM analysis of the Golub et al. (1999) data set (i.e. a SAM analysis for two class unpaired data) can be performed by

```
> sam.out <- sam(golub, golub.cl, rand = 123, gene.names = golub.gnames[,3])
> sam.out
```

SAM Analysis for the Two-Class Unpaired Case Assuming Unequal Variances

	Delta	p0	False	Called	FDR
1	0.1	0.5	2424.77	2739	0.44276
2	0.7	0.5	262.21	1248	0.10508
3	1.3	0.5	12.11	507	0.01195
4	1.8	0.5	0.74	210	0.00176
5	2.4	0.5	0.01	76	6.58e-05
6	3.0	0.5	0	15	0
7	3.6	0.5	0	5	0
8	4.1	0.5	0	2	0
9	4.7	0.5	0	2	0
10	5.3	0.5	0	0	0

The argument `rand` is set to 123 to make the results of `sam` reproducible.

The output of `sam.out` consists of

- **Delta:** the different values of Δ for which the numbers of genes and the estimated FDRs are computed,
- **p0:** the estimated prior probability that a gene is not differentially expressed,
- **False:** the number of falsely called genes (which is *not* the expected number of false positives, see below and Tusher et al., 2001),
- **Called:** the number of genes called differentially expressed,
- **FDR:** the estimated FDR computed by $p0 \cdot \text{False} / \text{FDR}$.

Note that the expected number of false positives is given by $p0 \times \text{False}$ such that the number of falsely called genes denoted by `False` is only equal to the expected number of false positives if $p0 = 1$.

If one would like to use the Wilcoxon rank sum statistic instead of a moderated t -statistic, the analysis can be done by

```
> sam.out2 <- sam(golub, golub.cl, method = wilc.stat, rand = 123)
```

A little bit more information about the former SAM analysis can be obtained by

```
> summary(sam.out)
```

SAM Analysis for the Two-Class Unpaired Case Assuming Unequal Variances

s0 = 0.0584 (The 0 % quantile of the s values.)

Number of permutations: 100

MEAN number of falsely called variables is computed.

	Delta	p0	False	Called	FDR	cutlow	cutup	j2	j1
1	0.1	0.5	2424.77	2739	0.44276	-0.177	0.228	1434	1747
2	0.7	0.5	262.21	1248	0.10508	-1.264	1.438	737	2541
3	1.3	0.5	12.11	507	0.01195	-2.299	2.488	311	2856
4	1.8	0.5	0.74	210	0.00176	-3.154	3.311	134	2976
5	2.4	0.5	0.01	76	6.58e-05	-4.157	4.259	44	3020
6	3.0	0.5	0	15	0	-5.577	5.139	4	3041
7	3.6	0.5	0	5	0	-Inf	5.971	0	3047
8	4.1	0.5	0	2	0	-Inf	7.965	0	3050
9	4.7	0.5	0	2	0	-Inf	7.965	0	3050
10	5.3	0.5	0	0	0	-Inf	Inf	0	3052

The last four columns of this table show the values of $\text{cut}_{\text{up}}(\Delta)$ and $\text{cut}_{\text{low}}(\Delta)$, which are the upper and lower cutoffs for a gene to be called differentially expressed, and i_1 and i_2 , which are the indices of the (ordered) genes specifying the cutoffs (for details, see Schwender, 2007).

The output of this analysis with **sam** contains a table of statistics for a set of initial values of Δ . If other values of Δ , let's say 1.5, 1.6, 1.7, ..., 2.4, are of interest, one can use **print** or **summary** to get the number of significant genes and the estimated FDR for these values of **delta**.

```
> print(sam.out, seq(1.5, 2.4, 0.1))
```



Figure 1: SAM plot for $\Delta = 2.4$.

SAM Analysis for the Two-Class Unpaired Case Assuming Unequal Variances

	Delta	p0	False	Called	FDR
1	1.5	0.5	4.48	377	0.005943
2	1.6	0.5	2.37	304	0.003899
3	1.7	0.5	1.49	262	0.002844
4	1.8	0.5	0.74	210	0.001762
5	1.9	0.5	0.43	191	0.001126
6	2.0	0.5	0.21	155	0.000678
7	2.1	0.5	0.12	132	0.000455
8	2.2	0.5	0.06	111	0.000270
9	2.3	0.5	0.03	98	0.000153
10	2.4	0.5	0.01	76	6.58e-05

The function `plot` can be used to generate a SAM plot for a specific value of `delta`.

```
> plot(sam.out, 2.4)
```

Note that the SAM plot is only generated if `delta` is a numeric value. If `delta` is a vector or `NULL`, a graphical representation of the table produced by `print`

is plotted.

The function `identify` makes it possible to obtain information about the genes by clicking on the SAM plot.

```
> identify(sam.out)
```

If `chip`, i.e. the chip name (e.g., "hgu133plus2"), is specified and `ll = TRUE` in `identify`, then the locus link and the symbol of the gene corresponding to the identified point are added to the output. For example, clicking on the point nearest to the upper right corner, i.e. the point corresponding to the gene with the largest positive expression score d , produces

	d.value	stdev	p.value	q.value	R.fold
M27891_at	8.1652	0.2958	0	0	7.2772

which does not contain the locus links since the chip type has not been specified.

If the chip name has been specified either by `chip` or by setting `data` to an `ExpressionSet` object, one can set `browse = TRUE` in `identify`. This opens the NCBI webpage corresponding to the Entrez / locus link of the gene identified by clicking on the SAM plot.

Gene-specific information about the genes called differentially expressed using a specific value of Δ (here $\Delta = 3.3$) can be obtained by

```
> sum.sam.out <- summary(sam.out, 3.3)
> sum.sam.out
```

SAM Analysis for the Two-Class Unpaired Case Assuming Unequal Variances

$s_0 = 0.0584$ (The 0 % quantile of the s values.)

Number of permutations: 100

MEAN number of falsely called variables is computed.

Delta: 3.3
cutlow: -Inf

```

cutup: 5.971
p0: 0.5
Identified Genes: 5
Falsely Called Genes: 0
FDR: 0

```

Identified Genes (using Delta = 3.3):

	Row	d.value	stdev	rawp	q.value	R.fold	Name
1	829	8.17	0.296	0	0	7.42	M27891_at
2	2124	7.96	0.178	0	0	3.68	X95735_at
3	2600	6.10	0.191	0	0	2.87	L09209_s_at
4	2664	5.98	0.392	0	0	6.46	Y00787_s_at
5	766	5.97	0.173	0	0	2.61	M16038_at

The generated table contains the row numbers of the identified genes in the data matrix used (**Row**), the values of the test statistic (**d.value**) and the corresponding standard deviations, i.e. the values of the denominator of this statistic (**stdev**), the unadjusted p-values (**rawp**), the q-values (**q.value**), the fold changes (**R.fold**), and the names of the identified genes as specified in **sam** by **gene.names** (**Name**).

By default, in the output of **summary** the identified variables are called genes (or SNPs if **method = cat.stat** is used). To change this, use, e.g.,

```
> print(sum.sam.out, varNames = "Proteins")
```

SAM Analysis for the Two-Class Unpaired Case Assuming Unequal Variances

```
s0 = 0.0584 (The 0 % quantile of the s values.)
```

```
Number of permutations: 100
```

```
MEAN number of falsely called variables is computed.
```

```

Delta: 3.3
cutlow: -Inf
cutup: 5.971
p0: 0.5
Identified Proteins: 5

```

Falsely Called Proteins: 0
FDR: 0

Identified Proteins (using Delta = 3.3):

	Row	d.value	stdev	rawp	q.value	R.fold	Name
1	829	8.17	0.296	0	0	7.42	M27891_at
2	2124	7.96	0.178	0	0	3.68	X95735_at
3	2600	6.10	0.191	0	0	2.87	L09209_s_at
4	2664	5.98	0.392	0	0	6.46	Y00787_s_at
5	766	5.97	0.173	0	0	2.61	M16038_at

if protein data is analyzed.

The rows of `golub` that contain the values of the differentially expressed genes can also be obtained by

```
> sum.sam.out@row.sig.genes
```

M16038_at	M27891_at	X95735_at	L09209_s_at	Y00787_s_at
766	829	2124	2600	2664

the general information about the set of significant genes by

```
> sum.sam.out@mat.fdr
```

	Delta	p0	False	Called	FDR	cutlow	cutup	j2	j1
1	3.3	0.5001357	0	5	0	-Inf	5.970848	0	3047

and the gene-specific information by

```
> sum.sam.out@mat.sig
```

	Row	d.value	stdev	rawp	q.value	R.fold
M27891_at	829	8.165222	0.2958251	0	0	7.422684
X95735_at	2124	7.964784	0.1778697	0	0	3.684479

L09209_s_at	2600	6.102371	0.1911219	0	0	2.872660
Y00787_s_at	2664	5.975750	0.3918749	0	0	6.455137
M16038_at	766	5.970848	0.1731333	0	0	2.606597

To obtain just the names of the genes called significant using $\Delta = 3.3$,

```
> list.siggenes(sam.out, 3.3)
```

```
[1] "M27891_at" "X95735_at" "L09209_s_at" "Y00787_s_at" "M16038_at"
```

If the value of Δ and thus the number of differentially expressed genes should be identified for which the FDR is controlled at a level of, say, 0.05, call

```
> findDelta(sam.out, fdr = 0.05)
```

The threshold seems to be at

	Delta	Called	FDR
5	0.926009	925	0.050024
6	0.926010	922	0.049851

If there is one value of Δ for which FDR is exactly equal to 0.05, only this value of Δ and the corresponding number of identified genes is returned. If there is no such Δ , then a lower and upper bound for Δ is shown. Here, we would use $\Delta = 0.926010$, as our goal is to find a value of Δ for which FDR is less than or equal to 0.05.

Similarly, the estimated FDR can be obtained for a specific number of identified genes. E.g., calling 200 genes differentially expressed leads to an estimated FDR of 0.00145.

```
> findDelta(sam.out, genes = 200)
```

	Delta	Called	FDR
5	1.859437	200	0.00145

5 Empirical Bayes Analysis of Microarrays

If a moderated test score such as a moderated t - or F -statistic should be used in the Empirical Bayes Analysis of Microarray, the optimal choice of the fudge factor a_0 has to be specified prior to the actual EBAM analysis by a standardized version of the EBAM analysis (for details, see Efron et al., 2001). This standardized analysis can be performed using the function `find.a0`.

Considering the Golub et al. (1999) data as provided by the package `multtest`, this analysis is thus done by

```
> find.out <- find.a0(golub, golub.cl, rand = 123)
```

where `rand` sets the random number generator in an reproducible state.

```
> find.out
```

EBAM Analysis for the Two-Class Unpaired Case Assuming Unequal Variances

Selection Criterion: Posterior ≥ 0.9

	a0	Quantile	Number	FDR
1	0.0000	-	668	0.0294
2	0.0584	0	655	0.0294
3	0.1183	0.2	617	0.0302
4	0.1456	0.4	604	0.0306
5	0.1789	0.6	590	0.0310
6	0.2270	0.8	561	0.0319
7	0.6528	1	479	0.0367

Suggested Choice for `a0`: 0

To obtain the number of identified genes and the estimated FDR for another than the default value of `delta`, i.e. the minimum posterior probability for a gene to be called differentially expressed, say `delta = 0.95`, call

```
> print(find.out, 0.95)
```


EBAM Analysis for the Two-Class Unpaired Case Assuming Unequal Variances

Selection Criterion: Posterior ≥ 0.95

	a0	Quantile	Number	FDR
1	0.0000	-	501	0.0147
2	0.0584	0	490	0.0146
3	0.1183	0.2	458	0.0151
4	0.1456	0.4	445	0.0153
5	0.1789	0.6	432	0.0156
6	0.2270	0.8	406	0.0160
7	0.6528	1	324	0.0190

Suggested Choice for a0: 0

Note that a gene is called differentially expressed if

- its posterior probability is larger than or equal to `delta`,
- there is no gene with a more extreme test score (i.e a larger value if the score is positive, or a smaller value if the score is negative) than the considered gene that is not called differentially expressed.

The output of `find.a0` suggest a value for the fudge factor a_0 . This suggestion is based – as proposed by Efron et al. (2001) – on the number of differentially expressed genes. (The value of a_0 is chosen that leads to the largest number of differentially expressed genes.)

One, however, should also consider the estimated FDR and take a look on the plot of the logit-transformed posterior probabilities for the different values of a_0 . The plot displayed in Figure 2 can be generated by

```
> plot(find.out)
```

After having determined the optimal choice for a_0 , the actual EBAM analysis with the choice suggested by `find.a0` can be done by

```
> ebam(find.out)
```

EBAM Analysis for the Two-Class Unpaired Case Assuming Unequal Variances



Figure 2: Logit-transformed posterior probabilities for standardized EBAM analyses using different values of the fudge factor a_0 .

Fudge Factor: $a_0 = 0$

	Delta	Number	FDR
1	0.9	698	0.0274

If one would like to use another value of a_0 , let's say the minimum, i.e. the 0% quantile, of the standard deviations of the genes, which is comprised by the second row of `find.out`, then the EBAM analysis can be performed by

```
> ebam(find.out, which.a0 = 2)
```

EBAM Analysis for the Two-Class Unpaired Case Assuming Unequal Variances

Fudge Factor: $a_0 = 0.0584$

	Delta	Number	FDR
1	0.9	724	0.0295

Since in `find.a0` a fast, crude estimate for the number of falsely called genes (i.e. the number of genes expected under the null hypothesis to have a posterior

probability larger than or equal to `delta`) is used instead of the exact number. If the analysis should directly start with calling `ebam`, it is thus necessary to set `fast = TRUE` to employ this crude estimate, and therefore, obtain the same results as with a combination of `find.a0` and `ebam`.

Thus,

```
> ebam(golub, golub.cl, a0 = 0, fast = TRUE, rand = 123)
```

EBAM Analysis for the Two-Class Unpaired Case Assuming Unequal Variances

Fudge Factor: a0 = 0

	Delta	Number	FDR
1	0.9	698	0.0274

leads to the same results as

```
ebam(find.out)
```

since $a_0 = 0$ is suggested by `find.out` as the optimal choice for the fudge factor.

The exact mean number of falsely called genes is used when `fast = FALSE`, which is the default setting for `fast`. Thus,

```
> ebam.out <- ebam(golub, golub.cl, a0 = 0, rand = 123)
```

leads to the results of the EBAM analysis employing the exact mean number of falsely called genes and the value for the fudge factor identified in the analysis with `find.a0`.

By default, the number of differentially expressed genes and the estimated FDR is computed for `delta = 0.9`. To get these statistics for other values of `delta`, say 0.91, 0.92, ..., 0.99, call

```
> print(ebam.out, seq(0.91, 0.99, 0.01))
```

EBAM Analysis for the Two-Class Unpaired Case Assuming Unequal Variances



Figure 3: Plot of the posterior probabilities from the actual EBAM analysis using $\delta = 0.9$ as threshold for a gene to be called differentially expressed.

Fudge Factor: $a_0 = 0$

	Delta	Number	FDR
1	0.91	703	0.0265
2	0.92	678	0.0237
3	0.93	641	0.0204
4	0.94	610	0.0175
5	0.95	567	0.0144
6	0.96	523	0.0117
7	0.97	472	0.0089
8	0.98	397	0.0059
9	0.99	293	0.0032

The posterior probabilities can be plotted by

```
> plot(ebam.out, 0.9)
```

where the threshold `delta` for a gene to be called differentially expressed has to be specified (see Figure 3).

Having selected a value for `delta`, say 0.99997, gene-specific information can be obtained by

```
> summary(ebam.out, 0.99997)
```

EBAM Analysis for the Two-Class Unpaired Case Assuming Unequal Variances

```
Delta: 0.99997
a0: 0
p0: 0.4851
Cutlow: -7.2768
Cutup: 9.7758
Identified Genes: 7
Estimated FDR: 0
```

Identified Genes (posterior \geq 0.99997):

	Row	z.value	posterior	local.fdr
1	2124	10.578	1	1.029e-06
2	829	9.776	1	2.953e-06
3	2939	-7.548	1	1.582e-05
4	1995	-7.416	1	1.980e-05
5	2386	-7.348	1	2.221e-05
6	717	-7.313	1	2.356e-05
7	2489	-7.277	1	2.507e-05

The generated table contains the row numbers of the identified genes in the data matrix (`Row`), the values of the test statistic (`z.value`), the posterior probabilities (`posterior`), and an adhoc estimate of the local FDR (`local.fdr`, see Efron et al., 2001).

Instead of employing `find.a0` to identify an optimal value of a_0 and using this value to compute a moderated t -statistic, it is also possible to use the ordinary t -statistic by setting $a_0 = 0$ in `find.a0`. (Since for the Golub et al. (1999) $a_0 = 0$ is the optimal choice for the fudge factor, this analysis has already been done with output `ebam.out`).

Note that actually Welch's t -statistic (assuming unequal group variances) is computed by default. If you would like to use the (moderated) t -statistic assuming equal group variances, call

```
> ebam(golub, golub.cl, a0 = 0, var.equal = TRUE, rand = 123)
```

EBAM Analysis for the Two-Class Unpaired Case Assuming Equal Variances

Fudge Factor: a0 = 0

	Delta	Number	FDR
1	0.9	712	0.0282

If you do not want to find the optimal value for the fudge factor, but would like to use a reasonable value for a_0 , say the median, i.e. the 50% quantile, of the standard deviations of the genes, call

```
> ebam(golub, golub.cl, quan.a0 = 0.5, rand = 123)
```

EBAM Analysis for the Two-Class Unpaired Case Assuming Unequal Variances

Fudge Factor: a0 = 0.1612

	Delta	Number	FDR
1	0.9	669	0.0296

Instead of using a moderated t -statistic, it is also possible to employ Wilcoxon rank sum statistic. Since in this case it is not necessary to specify a value for a_0 , an EBAM analysis with Wilcoxon rank sums is performed by

```
> ebam(golub, golub.cl, method = wilc.ebam, rand = 123)
```

EBAM Analysis for the Two-Class Unpaired Case Using Wilcoxon Rank Sums

	Delta	Number	FDR
1	0.9	736	0.0273

6 Writing your own Score Function

It is also possible to write your own function for computing other test scores as the one provided by `siggenes` (see Chapter 2).

For all three wrappers `sam`, `find.a0` and `ebam`, this function must have as input the two required arguments

data: A matrix or data frame containing the data. Each row of this data set should correspond to one of the m variables (e.g., genes), and each column to one of the n observations.

c1: A vector consisting of the class labels of the observations.

The function can also have additional optional arguments, i.e. arguments for which a default is specified, which can be called in `sam`, `find.a0` or `ebam`.

6.1 `sam`

The output of a function that should be used in `sam` must be a list consisting of the following objects:

d: A numeric vector containing the test scores of the genes.

d.bar: A numeric vector of length `na.exclude(d)` consisting of the sorted test scores expected under the null hypothesis.

p.value: A numeric vector of the same length and order as `d` containing the p -values of the genes.

vec.false: A numeric vector of the same length as `d` consisting of the one-sided expected numbers of falsely called genes, i.e. the mean numbers of test scores that are larger than the test scores of the genes if the test scores of the genes are positive, and the mean number of test scores smaller than the test scores of the genes if the test scores of the genes are negative. Let's, e.g., assume that the observed test scores are

```
c(2.4, -1.2, -0.8, 1.3)
```

and the test scores obtained by two permutations of the class labels are given by

```
c(1.4, 2.5, -1.4, -0.8, 2, 1.1, -0.9, -2.3)
```

then `vec.false` is given by

```
c(0.5, 1, 2, 1.5)
```

since, e.g., one value from the two permutations is larger than 2.4 (thus, $1/2$ is the one-sided expected number of falsely called genes), and two values are smaller than -1.2 (thus, $2/2$ is the one-sided expected number of falsely called genes).

s: A numeric vector containing the standard errors of the expression values. If no standard errors are available, set `s = numeric(0)`.

s0: A numeric value specifying the fudge factor. If the fudge factor is not computed, set `s0 = numeric(0)`.

mat.samp: A $B \times n$ matrix containing the permuted class labels. Set `mat.samp = matrix(numeric(0))` if the exact null distribution is known.

msg: A character vector containing messages that are displayed when the SAM specific S4 methods `print` and `summary` are called. Should end with `"\n\n"`. If no message should be shown, set `msg=""`

fold: A numeric vector containing the fold changes of the genes. Should be set to `numeric(0)` if another analysis than a two-class analysis is performed.

Assume, e.g., that we would like to perform a SAM analysis with the ordinary t -statistic assuming equal group variances and normality. The code of a function `t.stat` for such an analysis is given by

```
> t.stat <- function(data, cl){
+   require(genefilter) ||
+     stop("genefilter required.")
+   cl <- as.factor(cl)
+   row.out <- rowttests(data, cl)
+   d <- row.out$statistic
+   m <- length(na.exclude(d))
+   d.bar <- qt(((1:m) - 0.5)/m, length(cl) - 2)
+   p.value <- row.out$p.value
+   vec.false <- m * p.value/2
+   s <- row.out$dm/d
+   msg <- paste("SAM Two-Class Analysis",
+     "Assuming Normality\n\n")
+   list(d = -d, d.bar = d.bar, p.value = p.value,
+     vec.false = vec.false, s = s, s0 = 0,
+     mat.samp = matrix(numeric(0)),
+     msg = msg, fold = numeric(0))
+ }
```


where `row.out$dm` is a vector containing the numerators of the t -statistics.

Note that in the output of `t.stat` `d` is set to `-d`, since in `rowttests` the mean of group 2 is subtracted from the mean of group 1, whereas in `sam` with `method = d.stat` the difference is taken the other way around.

Now `t.stat` can be used in `sam` by setting `method = t.stat`:

```
> sam(golub, golub.cl, method = t.stat)
```

SAM Two-Class Analysis Assuming Normality

	Delta	p0	False	Called	FDR
1	0.1	0.496	2565.546	2808	0.453283
2	0.8	0.496	407.898	1426	0.141912
3	1.5	0.496	25.57	631	0.020104
4	2.2	0.496	0.755	220	0.001703
5	2.8	0.496	0.032	86	0.000182
6	3.5	0.496	0.001	39	1.76e-05
7	4.2	0.496	1.29e-05	9	7.09e-07
8	4.9	0.496	4.8e-09	1	2.38e-09
9	5.6	0.496	4.8e-09	1	2.38e-09
10	6.3	0.496	0	0	0

6.2 find.a0

A function that should be used as `method` in `find.a0` must return a list consisting of

r: A numeric vector containing the numerator of the observed test scores of the genes.

s: A numeric vector comprising the denominator of the observed test scores of the genes.

mat.samp: A matrix in which each row represents one of the permutations of the vector `cl` of the classlabels.

z.fun: A function that computes the values of the test statistics for each gene. This function must have the required arguments `data` and `cl`, but no further arguments. Its output must be a list consisting of two objects:

One called `r` that contains the numerator of the test statistic, and the other called `s` comprising the denominator of the test score. (This function is used to compute the test scores when considering the permuted class labels.)

In `find.a0`, both the observed and the permuted z -values are monotonically transformed such that the observed z -values follow a standard normal distribution. If the observed z -values, however, should follow another distribution, the output of the function that should be used in `find.a0` must also contain an object called

z.norm: A numeric vector of the same length as `r` containing the appropriate quantiles of the distribution to which the observed z -values should be transformed.

It is also possible to specify a header that is shown when printing the output of `find.a0` by adding the object

msg: A character vector containing the message that is displayed when the `FindA0` specific method `print` is called. Should end with `"\n\n"`.

For an example, let's assume that we would like to implement an alternative way for using a moderated version of the ordinary t -statistic in `find.a0`. This function based on `rowttests` from the packages `genefilter` could, e.g., be specified by

```
> t.find <- function(data, cl, B = 50){
+   require(genefilter)
+   z.fun <- function(data, cl){
+     cl <- as.factor(cl)
+     out <- rowttests(data, cl)
+     r<- out$dm
+     s<- r / out$statistic
+     return(list(r = -r, s = s))
+   }
+   mat.samp <- matrix(0, B, length(cl))
+   for(i in 1:B)
+     mat.samp[i, ] <- sample(cl)
+   z.out <- z.fun(data, cl)
+   msg <- paste("EBAM Analysis with a Moderated t-Statistic\n\n")
+ }
```

```
+      list(r = z.out$r, s = z.out$s,
+          mat.samp = mat.samp, z.fun = z.fun, msg = msg)
+ }
```

Note that in the output of `z.fun` `r` is set to `-r`, since in `rowttests` the mean of group 2 is subtracted from the mean of group 1, whereas in `find.a0` with `method = z.find` the difference is taken the other way around.

Now `t.find` can be employed in `find.a0` by setting `method = t.find`:

```
> t.out <- find.a0(golub, golub.cl, method = t.find, B = 100, rand =123)
> t.out
```

EBAM Analysis with a Moderated t-Statistic

Selection Criterion: Posterior ≥ 0.9

	a0	Quantile	Number	FDR
1	0.0000	-	645	0.0288
2	0.0606	0	630	0.0282
3	0.1292	0.2	595	0.0290
4	0.1562	0.4	585	0.0295
5	0.1849	0.6	575	0.0301
6	0.2272	0.8	561	0.0307
7	0.6418	1	473	0.0360

Suggested Choice for a0: 0

which produces the same results as

```
> find.a0(golub, golub.cl, var.equal = TRUE, rand =123)
```

EBAM Analysis for the Two-Class Unpaired Case Assuming Equal Variances

Selection Criterion: Posterior ≥ 0.9

	a0	Quantile	Number	FDR
1	0.0000	-	645	0.0288
2	0.0606	0	630	0.0282

3	0.1292	0.2	595	0.0290
4	0.1562	0.4	585	0.0295
5	0.1849	0.6	575	0.0301
6	0.2272	0.8	561	0.0307
7	0.6418	1	473	0.0360

Suggested Choice for `a0`: 0

and can also be used in the actual EBAM analysis:

```
> ebam(t.out)
```

EBAM Analysis with a Moderated t-Statistic

Fudge Factor: `a0` = 0

	Delta	Number	FDR
1	0.9	698	0.0277

6.3 ebam

A function that should be employed as `method` in `ebam` must output a list consisting of the objects

z: A numeric vector containing the test scores of the genes.

ratio: A numeric vector of the same length as `z` comprising the values of the ratio $f_0(z)/f(z)$, where f_0 is the distribution of the test scores expected under the null hypothesis, and f is the distribution of the observed test scores.

vec.pos: A numeric vector of the same length as `z` consisting of the numbers of test scores expected under the null hypothesis to be larger than or equal to $\text{abs}(z[i])$, $i = 1, \dots, \text{length}(z)$.

vec.neg: A numeric vector of the same length as `z` consisting of the numbers of test scores expected under the null hypothesis to be smaller than or equal to $-\text{abs}(z[i])$, $i = 1, \dots, \text{length}(z)$.

It is also possible to specify a header that is shown when `printing` or `summarizing` the output of `ebam` by adding the object

msg: A character vector containing the message that is displayed when the EBAM specific methods **print** or **summary** are called. Should end with "\n\n".

As an example, let's assume we would like to use the ordinary *t*-statistic in **ebam** and assume normality. A function for performing this type of analysis is, e.g., given by

```
> t.ebam<-function(data, cl){
+   require(genefilter)
+   cl <- as.factor(cl)
+   out <- rowttests(data, cl)
+   z <- -out$statistic
+   z.dens <- densspr(z)$y
+   m <- length(z)
+   vec.pos <- m * out$p.value / 2
+   z.null <- dt(z, length(cl) - 2)
+   msg<-paste("EBAM Analysis with t-Statistic Assuming Normality.\n\n")
+   list(z = z, ratio = z.null/z.dens, vec.pos = vec.pos,
+        vec.neg = vec.pos, msg = msg)
+ }
```

where **densspr** is a function for estimating a density based on a histogram and a Poisson regression (cf. Efron and Tibshirani, 1996, and the help file for **densspr**), and **vec.neg** = **vec.pos** since the null distribution is symmetric.

Now **t.ebam** can be used in **ebam** by setting **method** = **t.ebam**:

```
> ebam(golub, golub.cl, method = t.ebam)
```

EBAM Analysis with t-Statistic Assuming Normality.

	Delta	Number	FDR
1	0.9	718	0.0295

References

Efron, B., and Tibshirani, R. (1996). Using Specially Designed Exponential Families for Density Estimation. *Annals of Statistics*, 24, 2431–2461.

- Efron, B., and Tibshirani, R. (2002). Empirical Bayes Methods and False Discovery Rates for Microarrays. *Genetic Epidemiology*, 23, 70–86.
- Efron, B., Tibshirani, R., Storey, J. D., and Tusher, V. (2001). Empirical Bayes Analysis of a Microarray Experiment. *Journal of the American Statistical Association*, 96, 1151–1160.
- Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., and Lander, E. S. (1999). Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, 286, 531–537.
- Schwender, H. (2003). Assessing the False Discovery Rate in a Statistical Analysis of Gene Expression Data. *Diploma Thesis*, Department of Statistics, University of Dortmund.
- Schwender, H. (2007). Statistical Analysis of Genotype and Gene Expression Data. *Dissertation*, Department of Statistics, University of Dortmund.
- Schwender, H., Krause, A., and Ickstadt, K. (2003). Comparison of the Empirical Bayes and the Significance Analysis of Microarrays. *Technical Report*, University of Dortmund, Dortmund, Germany.
- Schwender, H., Krause, A., and Ickstadt, K. (2006). Identifying Interesting Genes with siggenes. *RNews*, 6(5), 45–50.
- Tusher, V. G., Tibshirani, R., and Chu, G. (2001). Significance Analysis of Microarrays Applied to the Ionizing Radiation Response. *Proceedings of the National Academy of Science*, 98, 5116–5121.