Differential expression with the limma package

Author: Georgios Asimomitis

The purpose of this exercise is to simulate some “microarray” data and explore how well different statistical tests separate truly differential expression.

Specifically, we will create a synthetic dataset with replicates from 2 experimental conditions and put in differential expression for some features. The goal is to see how well different statistical summaries can distinguish between those “truly” differential and those not differential.

```
library("limma")
```

```
## Warning: package 'limma' was built under R version 3.3.2
```

Next, we set some parameters for the simulation. You will modify these to explore a few situations.

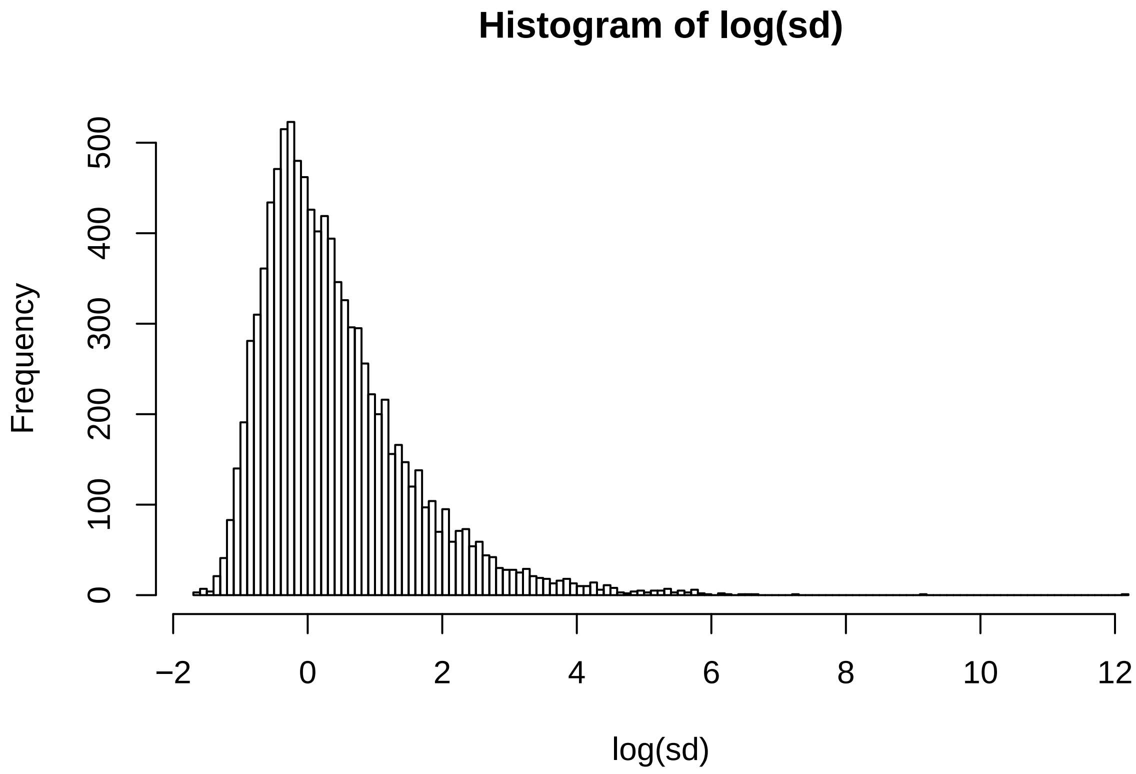
```
nGenes <- 10000                   # number of "features"
nSamples <- 6                     # number of samples (split equal in 2 groups)
pDiff <- .1                       # percent of genes "differential, 10%
grp <- rep(0:1,each=nSamples/2)   # dummy variable for exp. group
trueFC <- 2                       # log-fold-change of truly DE

d0 <- 1
s0 <- 0.8
sd = s0*sqrt(d0/rchisq(nGenes,df=d0))  # dist'n of s.d.
```

Distribution of “true” s.d

The computation of the “true” s.d is based on the inverse chi-square distribution. The corresponding histogram is the following one.

```
hist(log(sd),breaks=100);
```



Data Generation

The table of our data contains the gene expression information for 10000 genes over 6 different samples. We use normal distribution with standard deviation equal to "true" s.d, number of rows equal to the number of genes and number of columns equal to the number of samples.

```
y <- matrix(rnorm(nGenes*nSamples,sd=sd),
            nr=nGenes,nc=nSamples)
```

We add in "differential expression", randomly either in the positive or negative direction, to the first 1000 genes in samples 4,5,6:

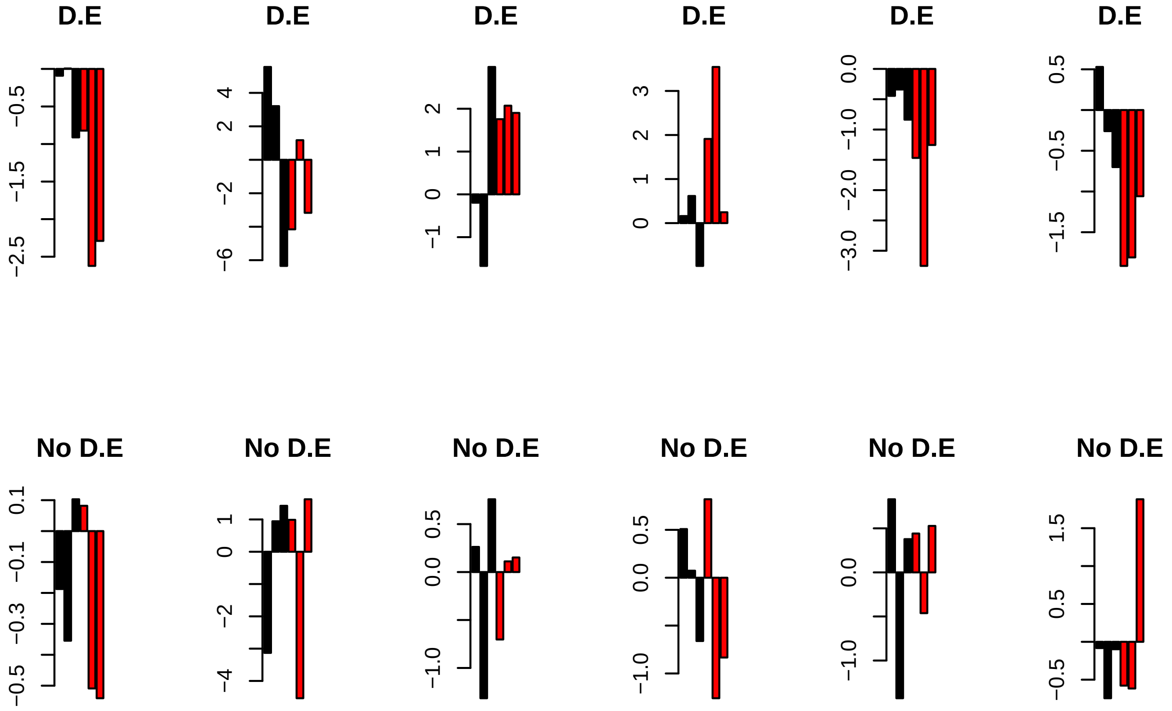
```
indD <- 1:floor(pDiff*nGenes)
diff <- sample(c(-1,1),max(indD),replace=TRUE)*trueFC
y[indD,grp==1] <- y[indD,grp==1] + diff
```

We visualize the difference in differential expression between the genes and the samples by barploting 6 random gene expressions from the first 1000 genes and 6 other from the last 9000 genes which are not differentialy expressed. Looking at the cases of the first 1000 genes we observe that the samples in which genes are differentially expressed (red) have larger positive or negative values comparing with the ones in which genes are not differentially expressed (black). The figures from the last 9000 genes display no difference between the values of gene expression between samples, as expected.

```r
par(mfrow=c(2,6))

barplot(y[3, ],col=as.factor(grp), main="D.E")
barplot(y[207, ],col=as.factor(grp),main="D.E")
barplot(y[677, ],col=as.factor(grp),main="D.E")
barplot(y[784, ],col=as.factor(grp),main="D.E")
barplot(y[827, ],col=as.factor(grp),main="D.E")
barplot(y[997, ],col=as.factor(grp),main="D.E")

barplot(y[1011, ],col=as.factor(grp),main="No D.E")
barplot(y[4006, ],col=as.factor(grp),main="No D.E")
barplot(y[6004, ],col=as.factor(grp),main="No D.E")
barplot(y[7001, ],col=as.factor(grp),main="No D.E")
barplot(y[8071, ],col=as.factor(grp),main="No D.E")
barplot(y[9574, ],col=as.factor(grp),main="No D.E")
```



Design Matrix

In general the design matrix contains the values of explanatory variables of a set of objects and is denoted by X in the linear model Y = Xb + e, where Y is the vector of observed data and b is the vector of parameters to estimate.The choice of design matrix is an important step in linear modeling as it encodes which coefficients will be fit in the model, as well as the inter-relationship between the samples.

In our case we construct a model in order to make comparisons between two different groups.Hence, the design matrice has two columns: an intercept column, which consists of 1?s, and a second column, which specifies which samples contain genes that are differentially expressed. The number of rows of the design matrix equals the number of samples. As a result, the values of the first 3 rows in the second column are 0, denoting the samples in which there is no differential expression, and the values of the last three rows are 1 denoting that the samples 4,5,6 contain differentially expressed genes. The form of the design matrix is in accordance with the parameters that need to be estimated. In our experiment b=transpose([b1 b2]), where b1 denotes the normal condition A where there is no differential expression and b2 denotes the condition of differential expression minus condition A.

The design matrix that feeds into limma is created as follows:

```
design <- model.matrix(~grp)
```

Model Construction

We use a standard limma pipeline to construct our model. LmFit fits a linear model for each gene given the data matrix y and the design matrix. eBayes takes as input the microarray linear model fit and computes moderated t-statistics, moderated F-statistics, and log-odds of differential expression by empirical Bayes moderation of the standard errors towards a common value.

```
fit <- lmFit(y,design)
fit <- eBayes(fit)
```

Classical t-test

For each row in the simulated table, we compute the classical 2-sample t-test using the built-in R function t.test.

```
classicalt = apply(y,1,function(x) t.test(x[1:3],x[4:6],paired = FALSE)$statistic);
```

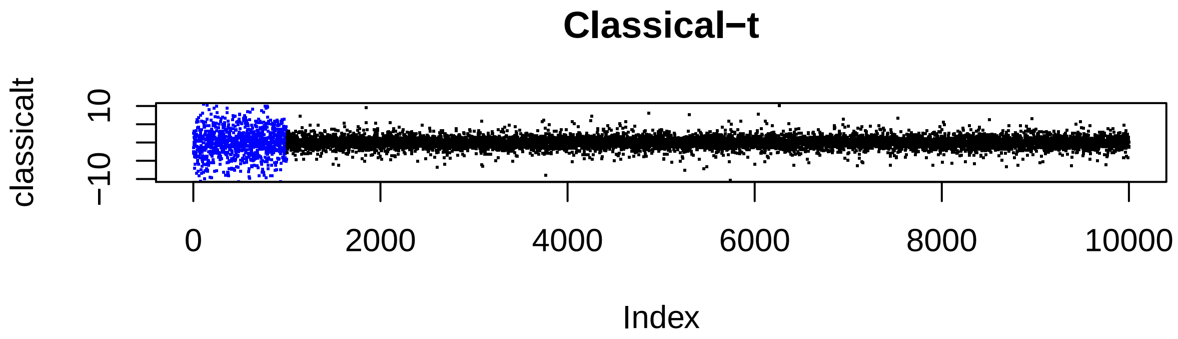
We can also calculate the classical t-test by using information that our constructed model provides. eBayes doesn't compute ordinary (unmoderated) t-statistics by default, but these can be easily extracted from the model output.

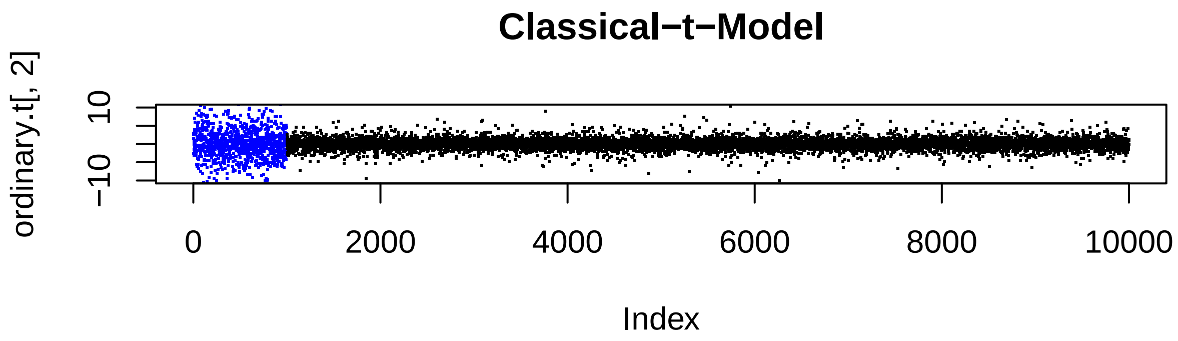
```
ordinary.t <- fit$coef / fit$stdev.unscaled / fit$sigma
```

The results of the classical t-test from both ways are presented below. The fact that they seem the same reveals that there is no difference if the t-test is computed out of the data matrix y or out of the output of our model.

```
cols <- rep("black",nrow(y))
cols[indD] <- "blue"

par(mfrow=c(2,1))
plot(classicalt, col=cols, ylim=c(-10,10), pch=".", main="Classical-t" )
plot(ordinary.t[ ,2], col=cols, ylim=c(-10,10), pch=".", main="Classical-t-Model" )
```

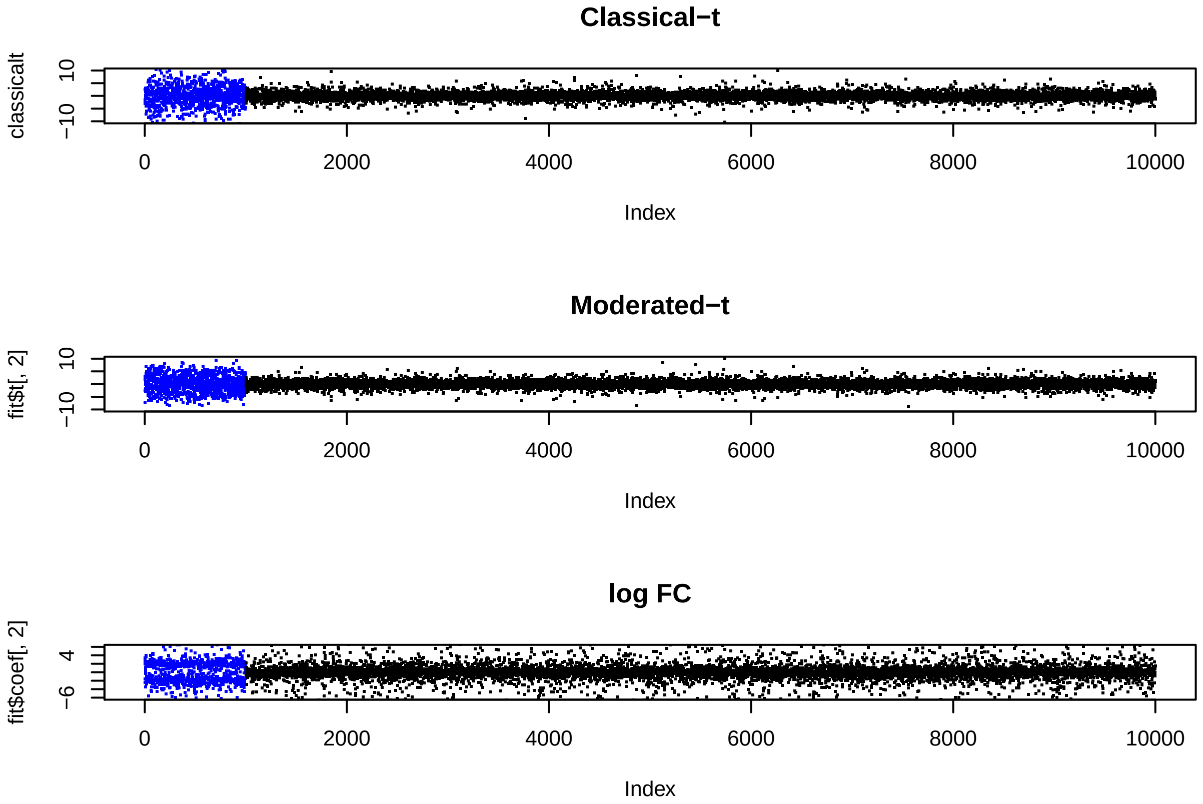




Three Statistical summaries

Together with the classical t-test we plot the moderated-t and the log FC statistics that are computed by eBayes.

```r
par(mfrow=c(3,1))
plot(classicalt, col=cols, ylim=c(-10,10), pch=".", main="Classical-t" )
plot( fit$t[,2], col=cols, ylim=c(-10,10), pch=".", main="Moderated-t" )
plot( fit$coef[,2], col=cols, ylim=c(-6,6), pch=".", main="log FC" )
```



For each gene, the classical t test displays the difference in mean between the group of samples with differential expression and the group of samples without d.e (2-sample t) relative to the variation in the sample data. Since var.equal is set to FALSE at t.test built in function, the variance is estimated separately for both groups. Out of the classical t-test plot we observe that t values are high (positively or negatively) in the first 1000 genes and around zero for the last 9000 genes. This is rational considering that we expect the difference in the mean of the two groups to be larger in the genes with differential expression.

In comparison with the classical t-test, the moderated t-test represents the difference in the mean of the two groups relative not only to the variance of each gene but also to the overall estimate of the variance. As a result, the variance used is a weighted average (each of the variances is weighted by the corresponding degrees of freedom).

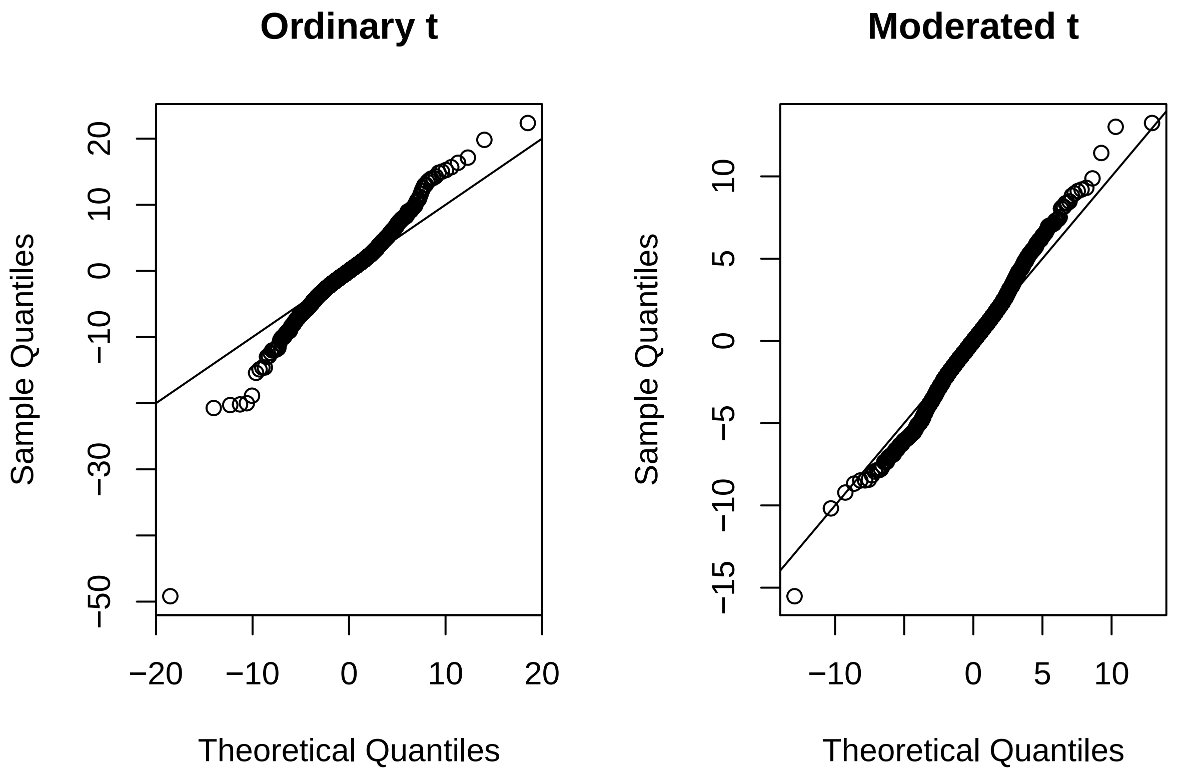
Within this frame of reference, the moderated t-statistic (t) computed by eBayes is the ratio of the log2-fold change to its standard error, which as already described, has been moderated across genes effectively borrowing information from the ensemble of genes to aid with inference about each individual gene. As a result, the distribution of the moderated t-values is more compact and takes into account the overall behaviour of variance. Classical t-test is more gene-specific since t values are computed separately of the overall variance behavior, thus it is more common to observe high (positive or negative) values. The moderated t-values are closer to each other, since their inference is not only based on gene variance, and offer a more secure indication of the difference between the two groups.

The third plot represents the log Fold Change for each gene and is a clear representation of the difference in the mean of the two groups. As a result in the first 1000 genes, in the majority of the cases, data is above or below zero, since the mean of gene expression for the two groups is not the same. However in the last 9000 genes in which there is no differential expression in all samples, data is distributed around zero as expected. Deviations from this general behavior is due to the random nature of the data generated initially.

Another useful way of comparing the classical t-test with the moderated t-test is by using the qq plot. As shown below the moderated data fall on the straight line whereas this is not the case in the classical t test.

Points out of the straight line denote that the sample quantiles in classical-test deviate from the theoretical quantiles of a Student's t distribution.

```
par(mfrow=c(1,2))
qqt(ordinary.t, df=fit$df.residual, main="Ordinary t")
abline(0,1)
qqt(fit$t, df=fit$df.total,main="Moderated t")
abline(0,1)
```



False Discovery Rate

By using benchmarkR we use the False Discovery Rate as a metric to compare the three methods. Function fdX takes as input the p values computed by each method and a vector that specifies the positives and negatives and computes the number of False Discoveries.

```
library(benchmarkR)
labels = rep(0,nrow(y))
labels[indD] = 1

re1 = SimResults(pval = log(fit$p.value[,2]),labels = labels)
```

```
## padj is missing, selected method (BH) is used to generate padj.
```

```
classicalt2 = apply(y,1,function(x) t.test(x[1:3],x[4:6],paired = FALSE)$p.value);
re2 = SimResults(pval = log(classicalt2),labels = labels)
```

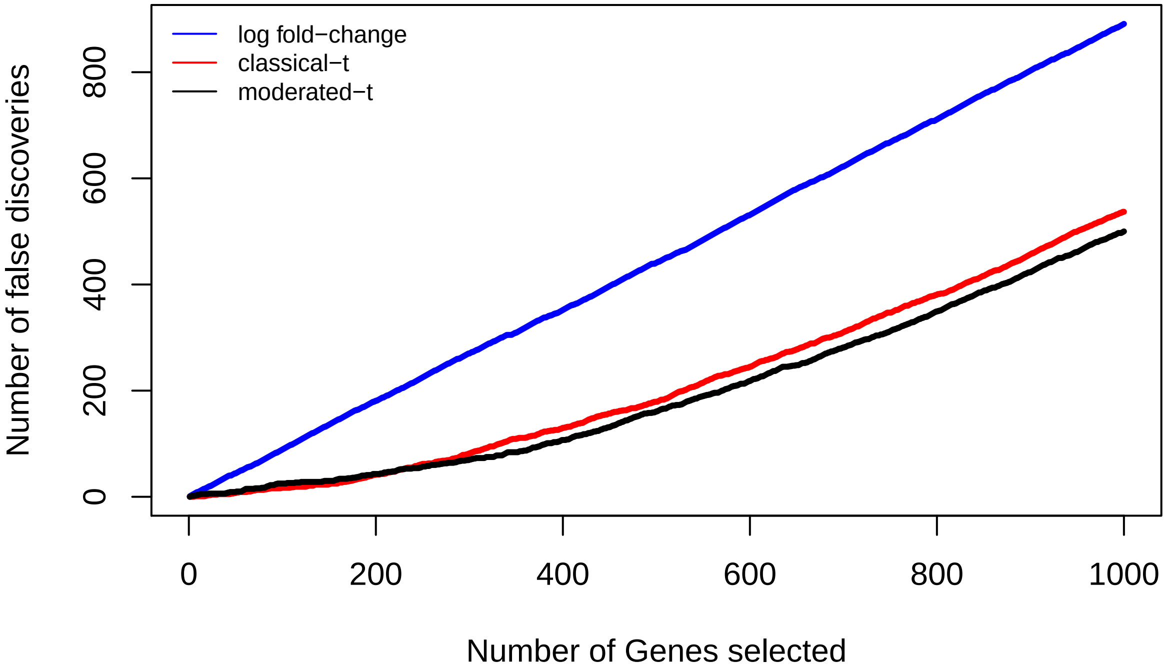
```
## padj is missing, selected method (BH) is used to generate padj.
```

```
re3 = SimResults(pval = log(abs(fit$coef[,1])),labels = labels)
```

```
## padj is missing, selected method (BH) is used to generate padj.
```

```
f1<-fdX(re3,col="blue",xlab="Number of Genes selected")
f2<-fdX(re2,col="red",xlab="Number of Genes selected",add=TRUE)
f3<-fdX(re1,col="black",xlab="Number of Genes selected",add=TRUE)

legend('topleft', c("log fold-change","classical-t","moderated-t") , lty=1, col=c('blue', 'red','black')
```



We observe that moderated-t test is slightly better than the classical t-test in terms of False Discovery Rate. This is rational if we consider also the distribution of t values in both methods. Given that p values are strongly connected with t values, we expect a low p value in the cases where t values are high. In classical t test there are some high t values (low p values) in cases where there is no differential expression. This cases probably correspond to False Discoveries. However, these cases are less in the moderated t test as values are more compact. In log fold change there are much more genes with high t values when there is no differential expression as no variance has been taken into account. As a result False Discoveries raise more.

Change number of samples to 40

We change the number of samples of 6 to 40.In the first 1000 genes the samples 21 to 40 are differentially expressed. We compute the t values for each statistical method and it seems that there is no such difference between the moderated and the classical t test as the distribution of data looks quite similar. This is demonstrated also in the False Discovery rate plot where the number of false discoveries for both methods is exactly the same. The red line is identical to the black one.

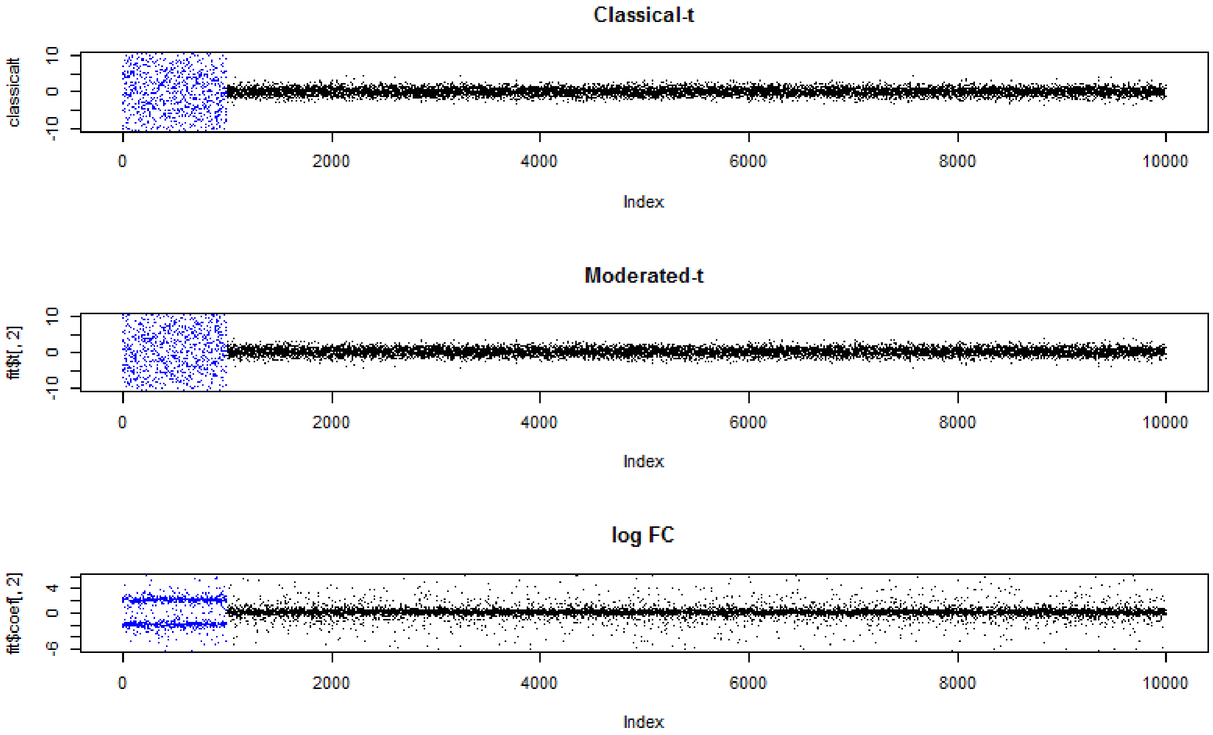


Figure 1: samples:40

Change number of genes d.e to 5000

Changing the number of genes with differential expression does not seem to affect the distribution of the t values for each type of test. Additionally, the number of false discoveries remains the same. Therefore, the behavior of the methods seems to remain unchanged when the number of d.e genes increases.

Change magnitude of the difference to 10

Changing the magnitude of the difference distributes the t values of the methods in higher scale. The Moderated t values and the classical ones seem quite similar. This is proved also from the fact that they have a similar false discovery rate even though the moderated method performs slightly better. What is significant is that the increase of the difference results to much lower number of false discoveries. This is because the means of the populations are more distant and thus it seems that it is easier for the tests to realise the difference between them. Therefore the number of false discoveries decreases.

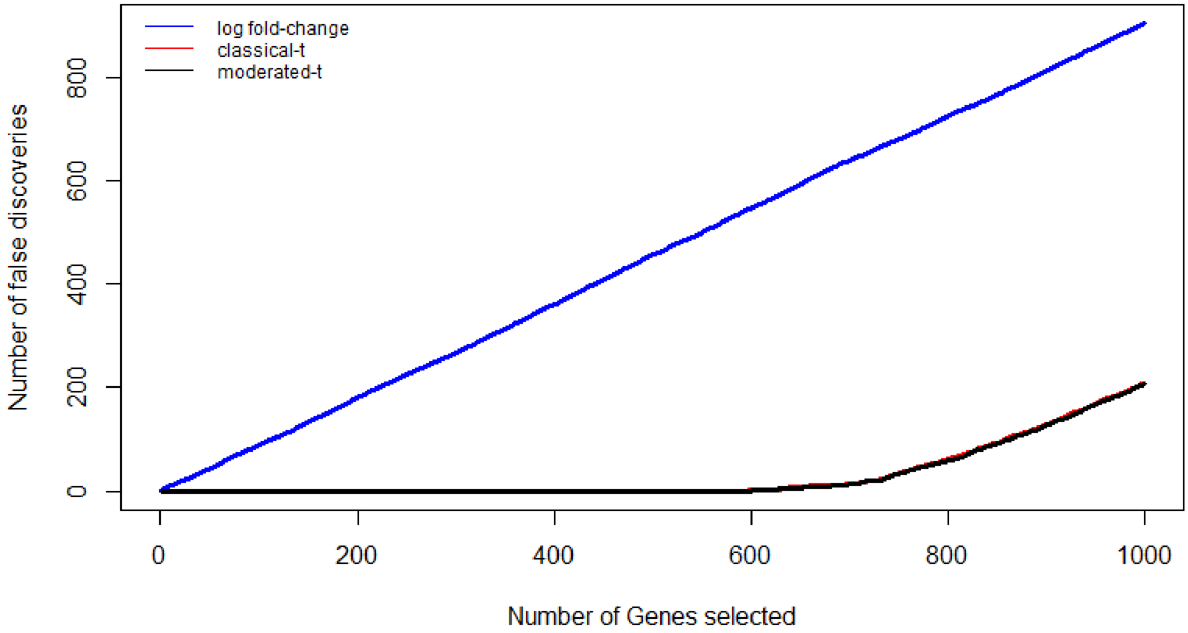


Figure 2: samples:40

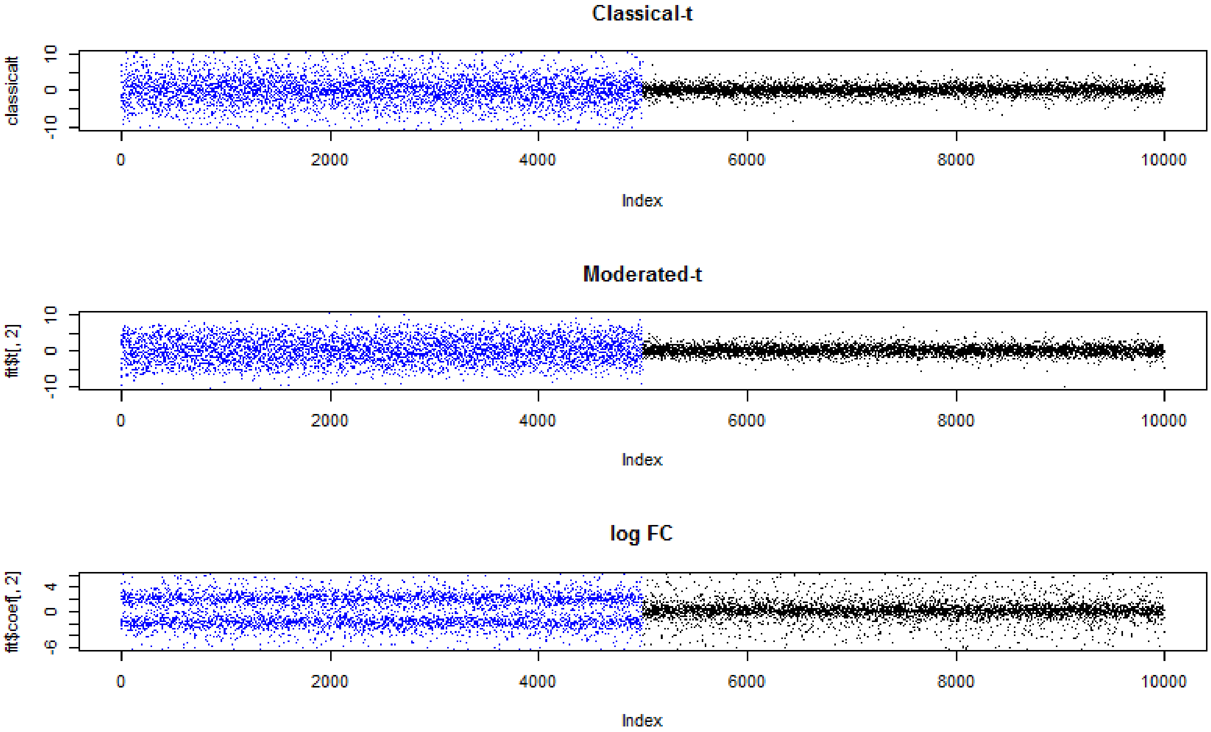


Figure 3: d.e genes:5000

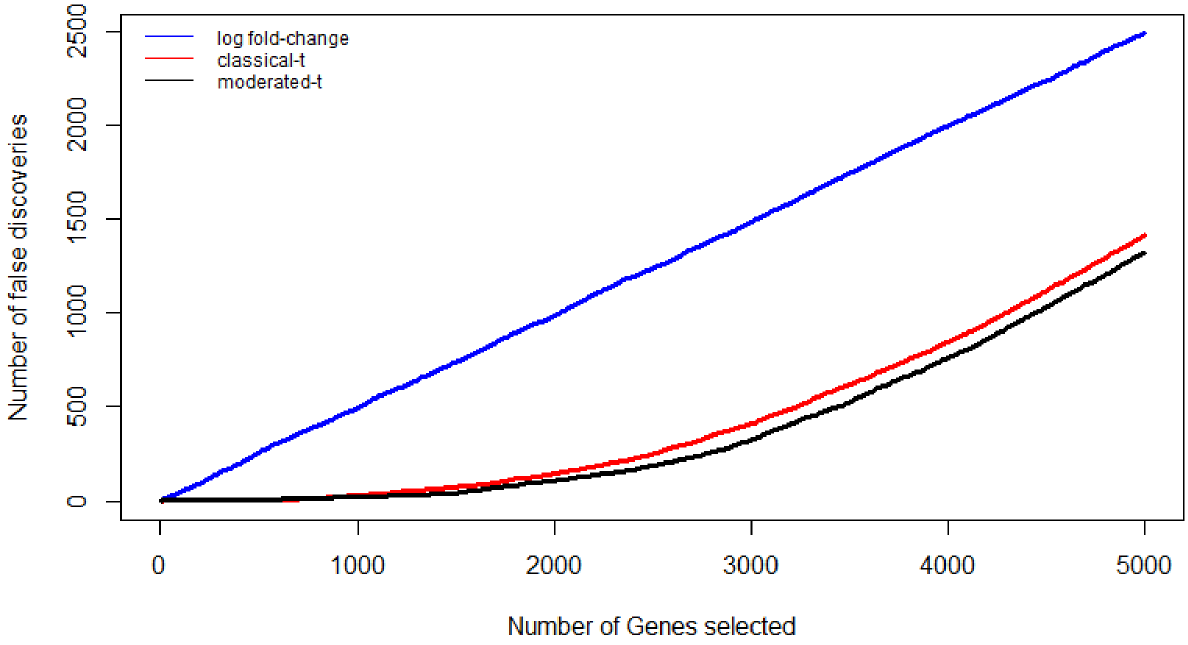


Figure 4: d.e genes:5000

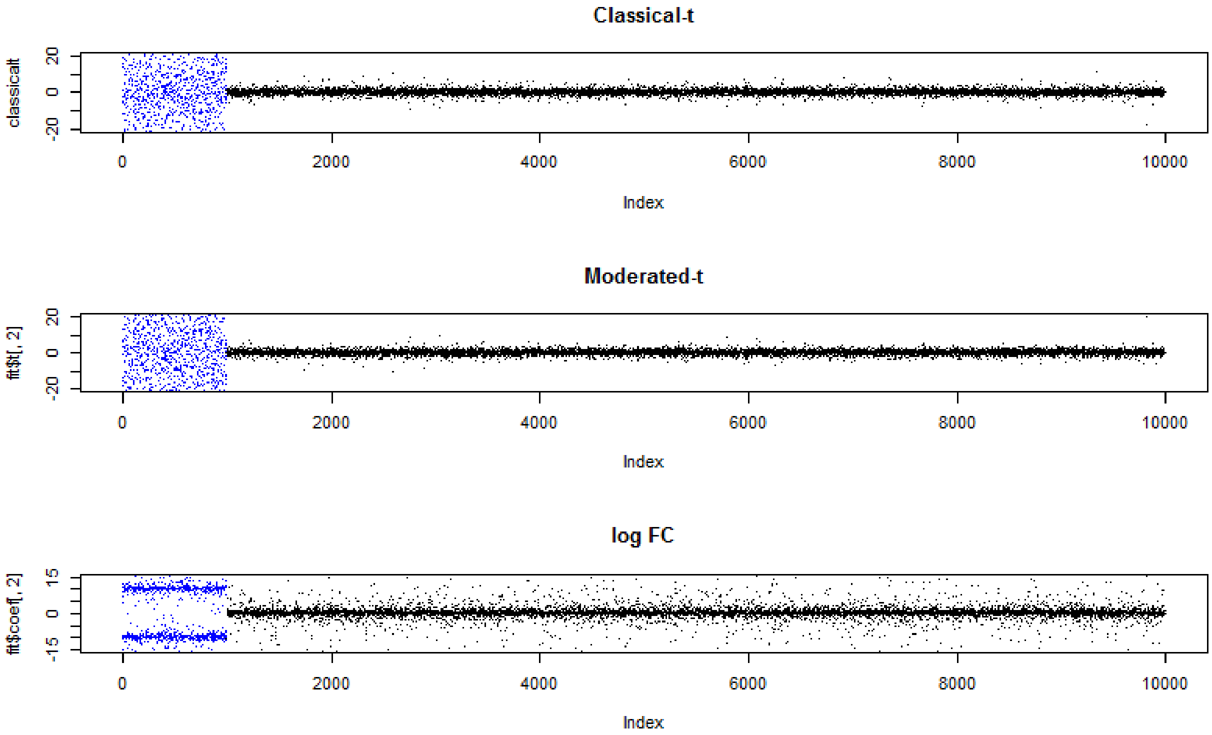


Figure 5: difference:10

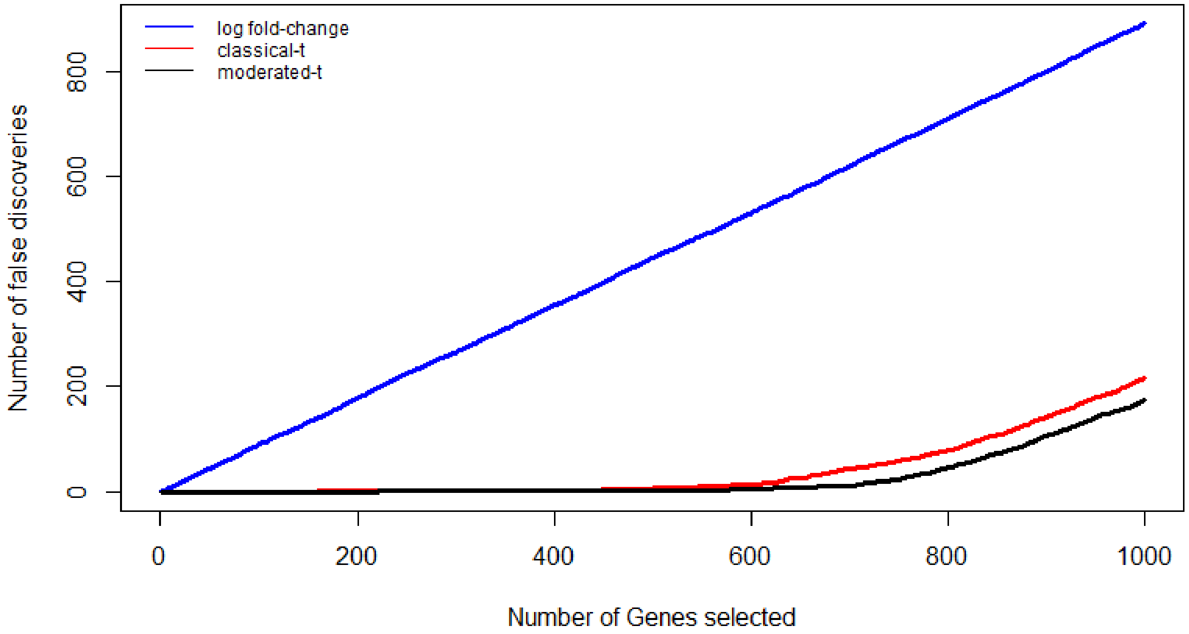


Figure 6: difference:10

Change s0 to 0.1

We decrease the pool variance to 0.1, a value which is close to zero. This affects the “true” s.d. Now that the variance of all genes is decreased, the contribution of the pooled variance to the moderated t-test is decreased too. Therefore, we expect that the behavior of the moderated t test will be closer to the classical t-test than it was before. In particular, we can observe their similarity in the plot of the three methods, as t values are similarly distributed. We can observe the similar behavior of these two methods clearer in the plot of the false discoveries where the two lines are identified for most of the genes selected. However, the moderated t test shows a slightly better performance as the pooled variance is not totally zero. What is also worth mentioning is the small number of false discoveries in comparison with the previous cases. The small pooled variance implies increased similarity in the values of gene expression between the genes. Therefore, the insertion of differential expression to the first 1000 genes creates a more clear distinction between the two groups.

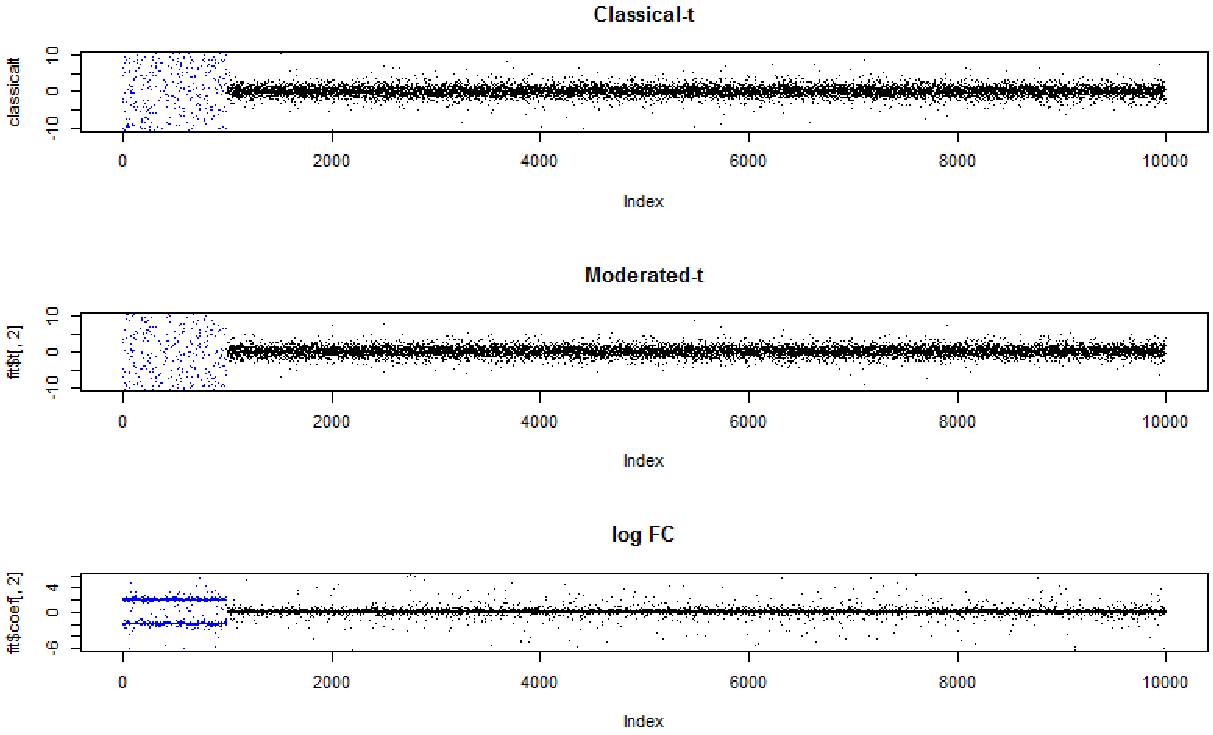


Figure 7: s0=0.1, “true” s.d change

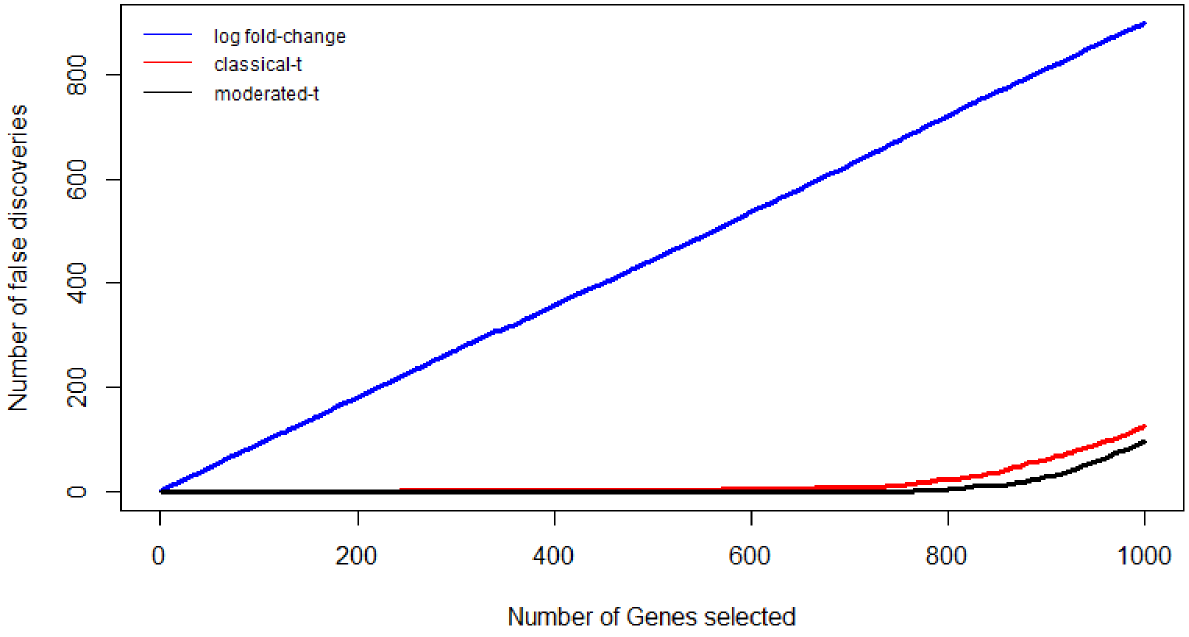


Figure 8: s0=0.1, “true” s.d change