Testing for differential expression

Wolfgang Huber

June 4, 2004

In this short exercise, we will explore two methods for selecting differentially expressed genes in a probe-by-probe manner. One method is based on hypothesis testing. We will use the t-test as a representant for this class of methods, which also comprises, for example, the Wilcoxon test and F-tests. Another method is based on the strength of discrimination, and is based on looking at properties of the Receiver Operating Characteristic Curves for each gene, if that gene were used as a discriminator.

First, let's load the necessary libraries and data.

```
> library(Biobase)
> library(ALL)
> data(ALL)
> library(multtest)
> library(genefilter)
```

We select the subset of B-cell ALLs whose molecular type is either *BCR/ABL* or *NEG*.

```
> sela <- intersect(grep("^B", as.character(ALL$BT)), which(as.character(ALL$mol.biol) %in%
+     c("BCR/ABL", "NEG")))
> ALLs <- ALL[, sela]
> table(ALLs$mol.biol)

ALL1/AF4  BCR/ABL E2A/PBX1      NEG   NUP-98  p15/p16
       0       37        0       42        0        0
```

Now select a subset of probes whose intensities were above 300 in at least 25% of the samples.

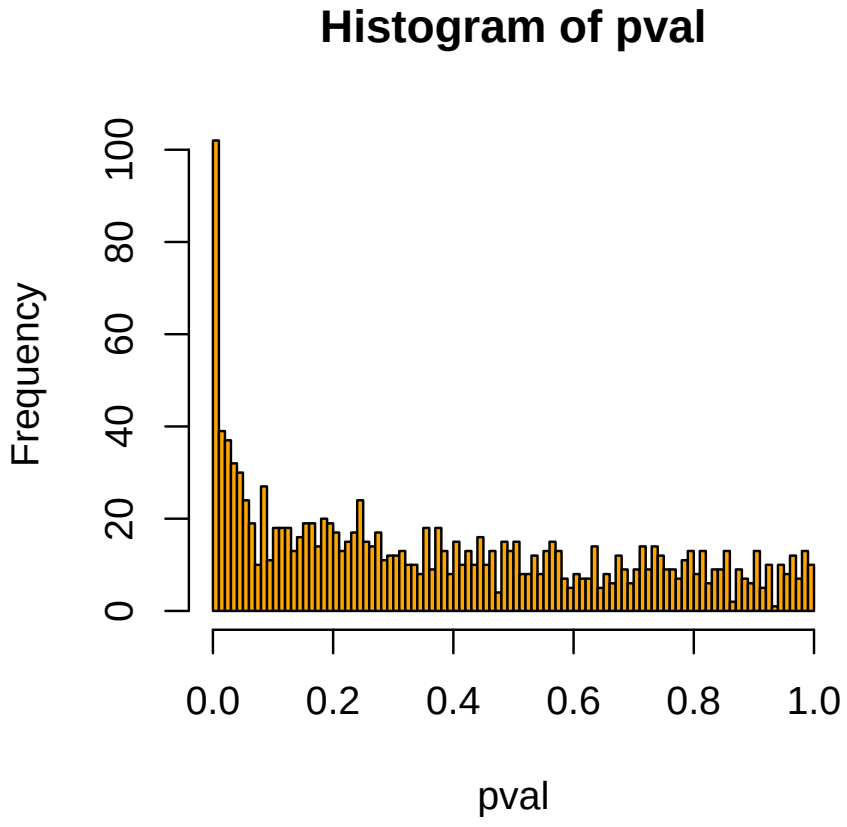
```
> f1 <- pOverA(0.25, log(300, 2))
> ff <- filterfun(f1)
> selp <- genefilter(ALLs, ff)
> ALLs <- ALLs[selp, ]
> sum(selp)

[1] 1362
```

Define a function that produces a logical vector out of an `exprSet`. The length of the vector is the number of samples, and the vector is intended to describe a grouping of the samples e.g. for running a t-test.

```
> classlabel <- function(x) {
+     stopifnot(class(x) == "exprSet", "mol.biol" %in% colnames(pData(x)))
+     return(as.numeric(pData(x)$mol.biol == "BCR/ABL"))
+ }
```

Now perform a probe-by-probe t-test and look at the histogram of resulting p-values:

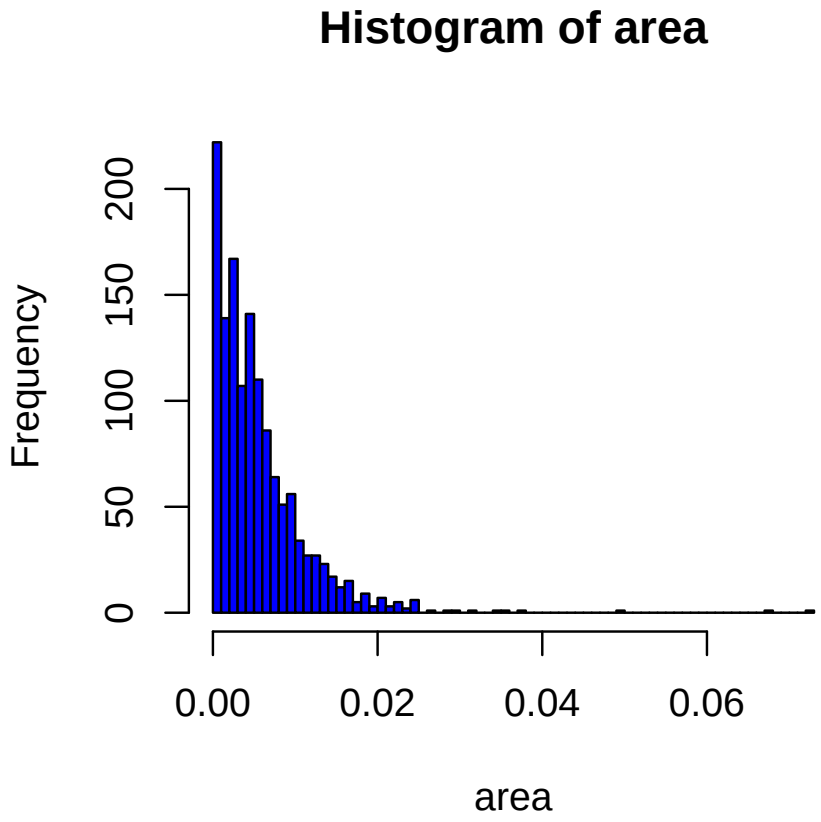
```
> table(classlabel(ALLs))

 0  1
42 37

> t <- mt.teststat(exprs(ALLs), classlabel = classlabel(ALLs),
+     test = "t.equalvar")
> pval <- 2 * (1 - pt(abs(t), df = ncol(exprs(ALLs) - 2)))
> hist(pval, breaks = 100, col = "orange")
```



Another way to select genes is by the *partial area under the curve* ($pAUC$), where the curve is a *receiver operating curve* (ROC).

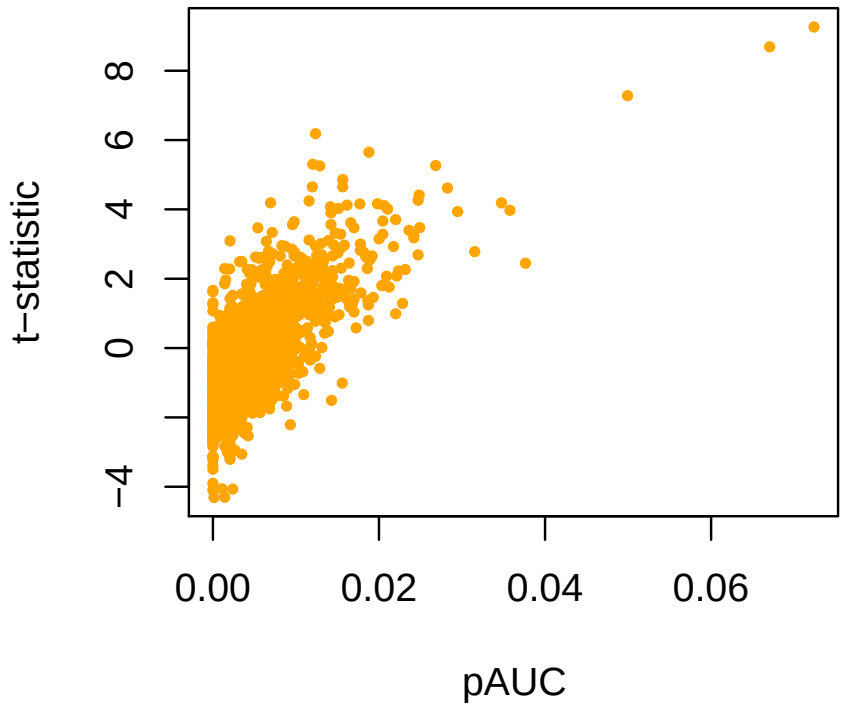
```
> library(ROC)
> my.pAUC <- function(x, p = 0.1, ...) {
```

```
+     roc <- rocdemo.sca(data = x, rule = dxrule.sca, ...)
+     return(pAUC(roc, p))
+ }
> area <- esApply(ALLs, 1, my.pAUC, truth = classlabel(ALLs))
> hist(area, breaks = 100, col = "blue")
```



The following plot shows that large pAUC values correspond to large values of the t-statistic, but not necessarily the other way round:

```
> plot(area, t, xlab = "pAUC", ylab = "t-statistic", pch = 16,
+     cex = 0.5, col = "orange")
```



Why are there no large values of `area` for very small values of `t`? What happens if you use

```
> area2 <- esApply(ALLs, 1, my.pAUC, truth = 1 - classlabel(ALLs))
```

As a strategy to select genes from a microarray experiment (e.g. for follow-up studies), one might think of at least two strategies:

1. choose all probes with a p-value less than some threshold (i.e. t-statistic larger than some threshold)
2. choose all probes with a pAUC larger than some threshold

Let's investigate how the number of probes we would select either way depends on the number of *samples* that we have. To this end, first define a wrapper function around `mt.teststat`.

```
> nrsel.ttest <- function(x, pthresh = 0.05) {
+     t <- mt.teststat(exprs(x), classlabel = classlabel(x), test = "t.equalvar")
+     pval <- 2 * (1 - pt(abs(t), df = ncol(exprs(ALLs) - 2)))
+     return(sum(pval < pthresh))
+ }
```

A similar wrapper around `my.pAUC`:

```
> nrsel.roc <- function(x, areathresh = 0.03) {
+     area <- esApply(x, 1, my.pAUC, truth = classlabel(x))
+     return(sum(area > areathresh))
+ }
```

However, resampling with that would become unbearably slow (someone needs to write a faster implementation of the functions `pAUX` and `rocdemo.sca`). So, for the purpose of this exercise, let's use a similar but slightly simpler criterion. The following function counts all those probes for which the fraction of samples in class 1 that has a value above the median of all samples is larger than larger than 0.75 or smaller than 0.25:

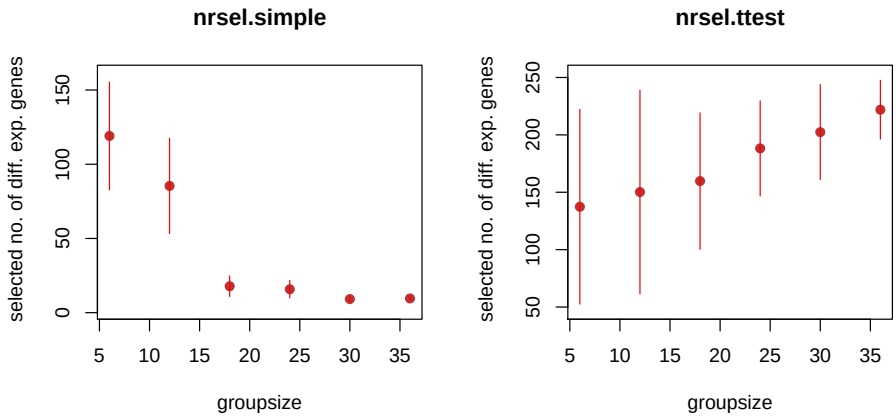
```
> nrsel.simple <- function(x, fracthresh = 0.25) {
+     rmed <- esApply(x, 1, median)
+     comp <- exprs(x) > rmed
+     nrs <- comp * matrix(classlabel(x), nrow = nrow(comp), ncol = ncol(comp),
+         byrow = TRUE)
+     nrs <- rowSums(nrs)
+     sizcl <- sum(classlabel(x))
+     return(sum(nrs >= (1 - fracthresh) * sizcl | nrs <= fracthresh *
+         sizcl))
+ }
```

Now let's write a function that does some resampling for various data set sizes:

```
> resample <- function(selfun, groupsize = seq(6, 36, by = 6),
+     nrep = 25) {
+     n <- matrix(nrow = nrep, ncol = length(groupsize))
+     for (i in seq(along = groupsize)) {
+         for (rep in 1:nrep) {
+             samplesubset <- c(sample(which(classlabel(ALLs) ==
+                 0), groupsize[i]), sample(which(classlabel(ALLs) ==
+                 1), groupsize[i]))
+             n[rep, i] <- do.call(selfun, args = list(x = ALLs[,
+                 samplesubset]))
+         }
+     }
+     mns <- apply(n, 2, mean)
+     stds <- apply(n, 2, sd)
+     plot(groupsize, mns, pch = 16, col = "#c03030", ylim = c(min(mns -
+         stds) - 5, max(mns + stds) + 5), xlab = "groupsize",
+         ylab = "selected no. of diff. exp. genes", main = selfun)
+     segments(groupsize, mns - stds, groupsize, mns + stds, col = "red")
+ }
```

And run it

```
> par(mfrow = c(1, 2))
> resample("nrsel.simple")
> resample("nrsel.ttest")
```



Which of the two criteria seems more reasonable?