Computer exercises on the BOOTSTRAP

Practical Microarray Analysis
Heidelberg, October 2003

Introduction

The most practical way to go through the exercise is to copy the program code into your R console window.

Clear your present workspace by using the command

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

Load the new data and needed packages:

```
load("BootStrap.RData")
require(affy)
```

The expression set to be used in this exercise is `Huang.RE` which is discussed in THE LANCET (2003) 361:1590-1596. The data contains microarrays of 52 women with breast cancer of whom 34 did not experience a recurrence of the tumour during a 3 years time period.

The data consists of VSN normalised expression measures which is summarised by median polish.

Exercise 1: Differential gene expression – one gene

In this exercise we repeat basic tests for a single gene which will be extended in exercise 2 to tests on the whole microarray. Gene No. 4402 with Affymetrix ID `"34361_at"` will be choosen.

```
rownames(exprs(Huang.RE))[4402]
```

Use the information on gene expression and recurrence to define the data for the two groups:

```
gene.exprs<- exprs(Huang.RE)[4402,]
names(pData(Huang.RE))
rec.info<-pData(Huang.RE)$Recurrence
table(rec.info)
gr.0<- gene.exprs[rec.info==0]
gr.1<- gene.exprs[rec.info==1]
```

A boxplot informs you about interesting features of the distribution of gene expression in both groups.

```
boxplot(gr.0,gr.1,names=c("No recurrence","Recurrence"))
```

Does the figure give convincing evidence for differential expression?

t-test

Calculate the t-test and try to understand the output

```
t.test(gr.1,gr.0)
```

Did you perform the t-test based on the pooled variance estimate or the Welsh-test? What is the difference between both tests? Use the command `?t.test` to figure out which t-test was performed?
Give the 95% confidence interval for the *log-transformed fold change* and calculate a 95% confidence interval for the *fold change*. The *fold change* expresses the change in gene expression for patients with a recurrence of the tumour compared to patients without a recurrence of the tumour. Is the situation suited to assume *log-transformed fold change*?

```
exp(t.test(gr.1,gr.0)$conf.int)
```

Try to give an interpretation of the result calculated.

Mann-Whitney test

The t-test is based on the assumption of normal distributed transformed gene expression in both groups. If one wants to drop this assumption, it is possible to perform a non-parametric test to compare the values measured in two independent groups: the *Mann-Whitney test*.

The *Mann-Whitney test* is developed to analyse the situation where the distribution of the values in one group is a shifted version of the distribution in the second group.

Calculate the *Mann-Whitney test* and try to understand the output.

```
wilcox.test(gr.0,gr.1)
```

The output does not give information on a 95% confidence interval for the shift parameter of interest. The following procedure helps you to calculate the confidence interval of interest.

```
delta.median<-median(gr.1)-median(gr.0)
alpha<-0.05
gr.star.0<-gr.0 + delta.median
gr.star.1<-gr.1
step<-0.001
#     Calculation of upper limit
upper<-0; p<-1
while (p>alpha)
      {
      p<-wilcox.test(gr.star.0, gr.star.1)$p.value
      gr.star.1<- gr.star.1+step; upper<-upper+step
      }
gr.star.1<-gr.1
#     Calculation of lower limit
lower<-0; p<-1
while (p>alpha)
      {
      p<-wilcox.test(gr.star.0, gr.star.1)$ p.value
      gr.star.1<- gr.star.1-step; lower <- lower –step
      }
conf.int.95<- delta.median+c(lower,upper)
conf.int.95.fold.change<-exp(conf.int.95)
```

Compare the 95% percent confidence interval calculated from the *Mann-Whitney test* with the confidence interval found via t-test. Can you give arguments which make the difference plausible?

Non-parametric permutation test

A statistical test needs a specific *Null-hypothesis*. In the case of comparing two groups the *Null-hypothesis* states no difference between both groups. Consequently, under the *Null-hypothesis*, both samples are treated as they would be generated from the same mechanism. It does not matter which of the observed values belongs to which group.

The *non-parametric permutation test* creates new groups by drawing values from the observed measurements without replacement and creating this way new group observations which fit the *Null-hypothesis*.

Using a special *test statistics* the *non-parametric permutation test* allows to estimate its distribution under the *Null-hypothesis of no difference*. The *p-value* for a two-sided test is calculated by determining the percentage of situation where the absolute value of the test statistics calculated from the groups with permutated values is at least as large as the absolute value of the test statistics calculated from the original groups.

In this exercise, the test statistics of interest will be the difference of *means* and the difference of *medians* between both groups.

First, two functions will be defined which calculate the test statistics of interest. The extension `.rfc` is used to denote an object as R-function.

```
delta.median.rfc<-function(g.1,g.0){return(median(g.1)-median(g.0))}
delta.mean.rfc<-function(g.1,g.0){return(mean(g.1)-mean(g.0))}
delta.mean.org<-delta.mean.rfc(gr.1,gr.0)
delta.median.org<-delta.median.rfc(gr.1,gr.0)
d.abs.mean.org<-abs(delta.mean.rfc(gr.1,gr.0))
d.abs.median.org<-abs(delta.median.rfc(gr.1,gr.0))
```

The objects `gene.exprs` and `rec.info` give the original observations. The values of both groups were determined by `gr.0<- gene.exprs[rec.info==0]`and `gr.1<- gene.exprs[rec.info==1]`. The permutation of the values observed is done by drawing `n.0 <- length(gr.0)` new values for group 0 from the observed expression measures without replacement. There are 52 observations of which 34 belong to group 0. Technically, the data for a permutation sample is created as follows:

```
ss.obs<-1:52
ind.gr.0<-sample(ss,n.0,replace=F)
gr.per.0<- gene.exprs[ind.gr.0]
gr.per.1<- gene.exprs[-ind.gr.0]
```

Now, it is possible to calculate the test statistics of interest for the permutated data set. Because the permutation step has to be repeated many times the above process will be summarised into a function.

The function uses as its first argument a counter i which is totally irrelevant to the test problem but needs to be introduced because of technical reasons - to perform the needed number of repetitions, resamples.

```
median.perm.rfc <-function(i,g.1,g.0)
            {
            n.0<-length(g.0)
            n.1<-length(g.1)
            ss<-1:(n.0+n.1)
            gg<-c(g.0,g.1)
            ind.gr.0<-sample(ss,n.0,replace=F)
            return(median(gg[-ind.gr.0])- median(gg[ind.gr.0]))
            }

mean.perm.rfc<-function(i,g.1,g.0)
            {
            n.0<-length(g.0)
            n.1<-length(g.1)
            ss<-1:(n.0+n.1)
            gg<-c(g.0,g.1)
            ind.gr.0<-sample(ss,n.0,replace=F)
            return(mean(gg[-ind.gr.0])- mean(gg[ind.gr.0]))
            }
```

Now it is possible to perform the resampling procedure. Because the p value depends on the number of repetitions, the choice of 1000 repetitions allows in general to calculate p values not below 0.001. The number of resample is crucial for the needed accuracy of the p-value.

```
median.perm.res<-unlist(lapply(1:1000,median.perm.rfc,g.0=gr.0,g.1=gr.1))
mean.perm.res<-unlist(lapply(1:1000,median.perm.rfc,g.0=gr.0,g.1=gr.1))
```

The *two-sided p-values* are now calculated by determining the percentage of situation where the absolute value of the test statistics calculated from the groups with permutated values is at least as large as the absolute value of the test statistics calculated from the original groups.

```
median.perm.p.value<-
sum(ifelse(abs(median.perm.res)>=d.abs.median.org,1,0))/1000
mean.perm.p.value<-sum(ifelse(abs(mean.perm.res)>=d.abs.mean.org,1,0))/1000
```

What is the exact meaning of the result 0?

Non-parametric bootstrap test and confidence intervals

The *non-parametric bootstrap* test follows similar principles as the *non-parametric permutation* test except the way how new data under the Null hypothesis is sampled. Instead of drawing without replacement, the *non-parametric bootstrap* test performs drawing with replacement.

This implies slight changes in the corresponding functions:

```
median.npboot.rfc <-function(i,values,group.ind)
            {
            ind.0<- group.ind==unique(group.ind)[1]
            ind.1<- group.ind==unique(group.ind)[2]
            values.new<- sample(values,length(values),replace=T)
            return(median(values.new[ind.1])- median(values.new[ind.0]))
            }
```

```
mean.npboot.rfc <-function(i,values,group.ind)
            {
            ind.0<- group.ind==unique(group.ind)[1]
            ind.1<- group.ind==unique(group.ind)[2]
            values.new<- sample(values,length(values),replace=T)
            return(mean(values.new[ind.1])- mean(values.new[ind.0]))
            }
```

Following the lines of the last section, the number of resamples is set to 1000.

```
median.npboot.res<-unlist(lapply(1:1000,median.npboot.rfc,
                                    values= gene.exprs, group.ind= rec.info))
mean.npboot.res<-unlist(lapply(1:1000,mean.npboot.rfc,
                                    values= gene.exprs, group.ind= rec.info))
```

The *two-sided p-values* are now calculated by determining the percentage of situation where the absolute value of the test statistics calculated from the groups with permutated values is at least as large as the absolute value of the test statistics calculated from the original groups.

```
median.npboot.p.value<-
            sum(ifelse(abs(median.npboot.res)>=d.abs.median.org,1,0))/1000
mean.npboot.p.value<-
            sum(ifelse(abs(mean.npboot.res)>=d.abs.mean.org,1,0))/1000
```

What is the exact meaning of the result 0?

The bootstrap can also be used to calculate confidence intervals for the mean or median difference by applying the sampling to the data of both groups seperately:

```
mean.npb.2g.rfc <-function(i,values,group.ind)
            {
            v.0<-values[group.ind==unique(group.ind)[1]]
            v.1<-values[group.ind==unique(group.ind)[2]]
            v.new.0<- sample(v.0,length(v.0),replace=T)
            v.new.1<- sample(v.1,length(v.1),replace=T)
            return(mean(v.new.1)- mean(v.new.0))
            }

median.npb.2g.rfc <-function(i,values,group.ind)
            {
            v.0<-values[group.ind==unique(group.ind)[1]]
            v.1<-values[group.ind==unique(group.ind)[2]]
            v.new.0<- sample(v.0,length(v.0),replace=T)
            v.new.1<- sample(v.1,length(v.1),replace=T)
            return(median(v.new.1)- median(v.new.0))
            }
```

Again, `lapply` is used to perform the repetitions.

```
median.npb.2g.res<-unlist(lapply(1:999,median.npb.2g.rfc,
                                    values=gene.exprs, group.ind=rec.info))
mean.npb.2g.res<-unlist(lapply(1:999,mean.npb.2g.rfc,
                                    values=gene.exprs, group.ind=rec.info))
```

In the above calculation only 999 resamples were performed. This is motivated by looking for 2.5% and 97.5% quantiles of the calculated bootstrap sample. From these quantiles the 95% confidence interval can be derived. By choosing the number of repetitions equal to 999 and

assuming that all resampled values will be different, then the 2.5% and 97.5% quantiles will not be determined by an value observed. This has several technical advantages. Additionally, one also performs a correction for a possible *bootstrap bias* by shifting the sample in the direction of the observed difference:

```
median.npb.95.ci<-quantile(median.npb.2g.res,probs=c(0.025,0.975))
median.shift<- delta.median.org-mean(median.npb.2g.res)
median.npb.95.ci<- median.npb.95.ci+ median.shift
mean.npb.95.ci<-quantile(mean.npb.2g.res,probs=c(0.025,0.975))
mean.shift<- delta.mean.org-mean(mean.npb.2g.res)
mean.npb.95.ci<- mean.npb.95.ci+ mean.shift
```

Use these 95% confidence intervals to calculate CI's for the *fold change* in gene expression between patients with and without recurrence.

```
exp(median.npb.95.ci)
exp(mean.npb.95.ci)
```

Parametric bootstrap test and confidence intervals

The *parametric bootstrap* test samples new data under the Null hypothesis by assuming a distribution described by parameters. In our example we would assume a common mean for both groups but the approach would allow to assume different variances in both groups. We have to modify our functions with respect to this idea:

```
mean.pb.rfc <-function(i,values,group.ind)
            {
            v.0<-values[group.ind==unique(group.ind)[1]]
            v.1<-values[group.ind==unique(group.ind)[2]]
            v.new.0<- rnorm(length(v.0),mean(c(v.0,v.1)),sd(v.0))
            v.new.1<- rnorm(length(v.1),mean(c(v.0,v.1)),sd(v.1))
            return(mean(v.new.1)- mean(v.new.0))
            }

median.pb.rfc <-function(i,values,group.ind)
            {
            v.0<-values[group.ind==unique(group.ind)[1]]
            v.1<-values[group.ind==unique(group.ind)[2]]
            v.new.0<- rnorm(length(v.0),mean(c(v.0,v.1)),sd(v.0))
            v.new.1<- rnorm(length(v.1),mean(c(v.0,v.1)),sd(v.1))
            return(median(v.new.1)- median(v.new.0))
            }
```

Now, the tests are performed using the same technical machinery as discussed in the section before:

```
median.pb.res<-unlist(lapply(1:1000,median.pb.rfc,
                                    values= gene.exprs, group.ind= rec.info))
mean.pb.res<-unlist(lapply(1:1000,mean.pb.rfc,
                                    values= gene.exprs, group.ind= rec.info))
median.pb.p.value<-
            sum(ifelse(abs(median.pb.res)>=d.abs.median.org,1,0))/1000
mean.pb.p.value<-
            sum(ifelse(abs(mean.pb.res)>=d.abs.mean.org,1,0))/1000
```

The calculation of the confidence intervals follows similar line as in the last section.
First, the functions performing a bootstrap in each group are defined:

```
mean.pb.2g.rfc <-function(i,values,group.ind)
            {
            v.0<-values[group.ind==unique(group.ind)[1]]
            v.1<-values[group.ind==unique(group.ind)[2]]
            v.new.0<- rnorm(length(v.0),mean(v.0),sd(v.0))
            v.new.1<- rnorm(length(v.1),mean(v.1),sd(v.1))
            return(mean(v.new.1)- mean(v.new.0))
            }

median.pb.2g.rfc <-function(i,values,group.ind)
            {
            v.0<-values[group.ind==unique(group.ind)[1]]
            v.1<-values[group.ind==unique(group.ind)[2]]
            v.new.0<- rnorm(length(v.0),mean(v.0),sd(v.0))
            v.new.1<- rnorm(length(v.1),mean(v.1),sd(v.1))
            return(median(v.new.1)- median(v.new.0))
            }
```

Then, the bootstrap is performed and the quantile method is used to calculöate the confidence interval.

```
median.pb.2g.res<-unlist(lapply(1:999,median.pb.2g.rfc,
                                    values=gene.exprs, group.ind=rec.info))
mean.pb.2g.res<-unlist(lapply(1:999,mean.pb.2g.rfc,
                                    values=gene.exprs, group.ind=rec.info))

median.pb.95.ci<-quantile(median.pb.2g.res,probs=c(0.025,0.975))
median.shift<- delta.median.org-mean(median.pb.2g.res)
median.pb.95.ci<- median.pb.95.ci+ median.shift
mean.pb.95.ci<-quantile(mean.pb.2g.res,probs=c(0.025,0.975))
mean.shift<- delta.mean.org-mean(mean.pb.2g.res)
mean.pb.95.ci<- mean.pb.95.ci+ mean.shift
```

Use these 95% confidence intervals to calculate CI's for the *fold change* in gene expression between patients with and without recurrence.

```
exp(median.pb.95.ci)
exp(mean.pb.95.ci)
```

Compare the 95% CIs for the mean created by the t-test, the non-parametric, and the parametric bootstrap.

Compare the 95% CIs for the median created by the Mann-Whitney test, the non-parametric, and the parametric bootstrap.

Exercise 2: Differential gene expression – the whole array

In the following exercise, the techniques applied for one gene will be extended to the whole array. The techniques available in *bioconductor* to perform this analysis will be presented. An additional package – *multtest* - has to be loaded:

```
require(multtest)
```

t-test

The value of the t-statistic can be simultaneously calculated as follows

```
Huang.t.stat.res<-mt.teststat(exprs(Huang.RE),pData(Huang.RE)$Recurrence)
```

Use the command `?mt.teststat` to figure out which t-test was performed?

There are 52 patients in two *independent* groups resulting in 50 *degrees of freedom* (DF) for the t-test. The corresponding *two-sided* p-values can now be calculated by

```
Huang.t.stat.p.values<- (1-pt(abs(Huang.t.stat.res),50))*2
```

The array contains 12625 probe sets. Looking for important genes which have a good chance to be differentially expressed between both groups based on simple testing needs to adjust the p-values for the fact of multiple testing. Basic procedures are given in the function `mt.rawp2adjp`.

```
Huang.t.stat.p.adj<- mt.rawp2adjp(Huang.t.stat.p.values)
```

To see the results of the 40 *strongest* genes look at the first component of the object which is a matrix and choose the first 40 lines:

```
Huang.t.stat.p.adj[[1]][1:40,]
```

The results show many genes with potential differential expression even after adjustment for multiple testing.

In order to find the Affymetrix IDs for the genes which will be differentially expressed and requiring a FDR below 0.01 one would choose all genes corresponding to the lines in `Huang.t.stat.p.adj[[1]]` where the values in the last column are not above 0.01. The values given in the last column are calculated using the procedure of *Benjamini and Yekutieli* which generalises the procedure of *Benjamini and Hochberg* to the case where the expression of genes is not assumed to be independent from each other.

```
ind.gene.names<-
Huang.t.stat.p.adj[[2]][Huang.t.stat.p.adj[[1]][,"BY"]<=0.01]
Huang.t.stat.gene.BY.01.res<-dimnames(exprs(Huang.RE))[[1]][ind.gene.names]
```

Mann-Whithey test

The above procedure can also be performed for the Mann-Whitney test:

```
Huang.MW.stat.res<-mt.teststat(exprs(Huang.RE),pData(Huang.RE)$Recurrence,
test="wilcoxon")
```

If `` `test="wilcoxon"' `` is used, the tests are based on standardised rank sum Wilcoxon statistics. Therefore, the p-values can be calculated using the normal distribution.

```
Huang.MW.p.values<- (1-pnorm(abs(Huang.MW.stat.res)))*2
```

In order to find the genes of interest we follow the procedure above. But we will take the FDR less strict and use 5% to compensate for the loss of information when using a non-parametric test.

```
Huang.MW.p.adj<- mt.rawp2adjp(Huang.MW.p.values)
ind.gene.names<- Huang.MW.p.adj[[2]][Huang.MW.p.adj[[1]][,"BY"]<=0.05]
Huang.MW.gene.BY.05.res<-dimnames(exprs(Huang.RE))[[1]][ind.gene.names]
```

There a considerable overlay in the genes discovered:

```
Huang.MW.gene.BY.01.res[Huang.MW.gene.BY.05.res%in%Huang.t.stat.gene.BY.01.res]
```

Non-parametric permutation test

The ideas presented in exercise 1 can be translated to the setting of the array. This will be a computing intensive activity and may take some time even on powerful machines. For didactic reasons, a smaller expression set will be used, containing the first 200 *probe sets* of the original arrays. The new expression set is called `H.200.RE`.

```
H.200.RE<-new("exprSet")
exprs(H.200.RE)<-exprs(Huang.RE)[1:200,]
pData(H.200.RE)<-pData(Huang.RE)
```

The *multtest* library offers a very efficient procedure to perform permutations on the expression values of a probe set. Permutation can be performed for a list of inbuild statistics. The default number of permutations is set to 10000. In the given situation there exist ~ 4.3e+13 possible permutations.

The following function calculates for each probe set the permutation p-value based on the t-statistics (Welch):

```
perm.Welch.rfc<-function(values,group.ind,anz.perm=10000)
            {
            v.0<-values[group.ind==unique(group.ind)[1]]
            v.1<-values[group.ind==unique(group.ind)[2]]
            se.org<-sqrt(var(v.1)/length(v.1)+ var(v.0)/length(v.0))
            welch.org<-(mean(v.1)-mean(v.0))/se.org
            welch.per<- mt.sample.teststat(values,group.ind,B=anz.perm)
            return(sum(ifelse(abs(welch.per)>abs(welch.org),1,0))/ anz.perm)
            }
```

All permutation p-values for the reduced expression set `H.200.RE` are calculated using the following command:

```
H.200.RE.Welch.perm.p.values<-apply(exprs(H.200.RE),1,perm.Welch.rfc,
                                    group.ind=rec.info)
```

or alternatively

```
H.200.RE.Welch.perm.p.values <-
                  esApply(H.200.RE,1,perm.Welch.rfc,group.ind=rec.info)
```

The calculation on my older laptop took 3 minutes. The calculation of permutation p-values for the whole array with 12625 probe set would need around 190 minutes.

The vector `H.200.RE.Welch.perm.p.values` can be used as an argument to the function `mt.rawp2adjp` to adjust the p-values for multiple testing.

Non-parametric bootstrap test and confidence intervals

The multtest library does not offer procedures for the bootstrap approach. Such a procedure has to be programmed by ourselves (or taken from other packages like boot). The following function returns bootstrap p-values for the difference of medians. The core of the following calculation consists of functions which were defined in exercise 1:

```
median.npboot.rfc <-function(i,values,group.ind)
            {
            ind.0<- group.ind==unique(group.ind)[1]
            ind.1<- group.ind==unique(group.ind)[2]
            values.new<- sample(values,length(values),replace=T)
            return(median(values.new[ind.1])- median(values.new[ind.0]))
            }
```

This function will be used to calculate the bootstrap p-value based on 1000 resamples as follows:

```
median.npboot.p.value.rfc<- function(values,group.ind,anz.boot=1000)
            {
            v.0<-values[group.ind==unique(group.ind)[1]]
            v.1<-values[group.ind==unique(group.ind)[2]]
            d.median.org<-(median(v.1)-median(v.0))
            d.median.boot<-lapply(1:anz.boot,median.npboot.rfc,
                                  values=values,group.ind=group.ind)
            d.median.boot<- unlist(d.median.boot)
            p.value<- sum(ifelse(abs(d.median.boot)>=abs(d.median.org),1,0))/
                              anz.boot
            return(p.value)
            }
```

Applied to our reduced array we get

```
H.200.RE.median.diff.npb.p.values <-
            esApply(H.200.RE,1, median.npboot.p.value.rfc,group.ind=rec.info,
                                                    anz.boot=1000)
```

This calculation took 4 minutes on my computer. Increasing the number of bootstrap samples would need more time, extension to the whole array would also add computing load. Such calculation make it necessary to parallelise the computation by distributing the 63 packages of expression sets a 200 probe sets on a network of computers or on a network of several processors. The R environment offers such possibilities.

Finally, the bootstrap will be used to calculate 95% confidence intervals for the difference of medians. Using a function written in exercise 1 we follow the procedure described above:

```
median.npb.2g.rfc <-function(i,values,group.ind)
            {
            v.0<-values[group.ind==unique(group.ind)[1]]
            v.1<-values[group.ind==unique(group.ind)[2]]
            v.new.0<- sample(v.0,length(v.0),replace=T)
            v.new.1<- sample(v.1,length(v.1),replace=T)
            return(median(v.new.1)- median(v.new.0))
            }
```

```
median.npboot.ci.rfc<- function(values,group.ind,anz.boot=999,alpha=0.05)
            {
            lower<-alpha/2
            upper<-1-alpha/2
            v.0<-values[group.ind==unique(group.ind)[1]]
            v.1<-values[group.ind==unique(group.ind)[2]]
            d.median.org<-(median(v.1)-median(v.0))
            npb.res<-unlist(lapply(1: anz.boot,median.npb.2g.rfc,
                                   values=values, group.ind=group.ind))
            npb.95.ci<-quantile(npb.res,probs=c(lower,upper))
            shift<- d.median.org-median(npb.res)
            npb.95.ci<- npb.95.ci+ shift
            return(c(lower= npb.95.ci[1],upper= npb.95.ci[2]))
            }
```

The 95% CI for the probe sets on the reduced expression set are given by

```
H.200.RE.median.diff.npb.ci <-
            esApply(H.200.RE,1, median.npboot.ci.rfc,group.ind=rec.info,
                                        anz.boot=999,alpha=0.05)
```

The object `H.200.RE.median.diff.npb.ci` is a 2 x 200 matrix, the list of 95% confidence intervals for the fold change of all probe sets in the reduced array are given by

```
exp(t(H.200.RE.mean.diff.npb.ci))
```

where t() transposes the 2 x 200 matrix to a 200 x 2 matrix.

Parametric bootstrap test and confidence intervals

The ideas presented for the non-parametric bootstrap can be analogously formulated for the parametric bootstrap.

```
median.pb.rfc <-function(i,values,group.ind)
            {
            v.0<- values[group.ind==unique(group.ind)[1]]
            v.1<- values[group.ind==unique(group.ind)[2]]
            v.new.0<- rnorm(length(v.0),mean(c(v.0,v.1)),sd(v.0))
            v.new.1<- rnorm(length(v.1),mean(c(v.0,v.1)),sd(v.1))
            return(median(v.new.1)- median(v.new.0))
            }
```

The function to calculate probe set wise bootstrap p-values is

```
median.pb.p.value.rfc<- function(values,group.ind,anz.boot=1000)
            {
            v.0<-values[group.ind==unique(group.ind)[1]]
            v.1<-values[group.ind==unique(group.ind)[2]]
            d.median.org<-(median(v.1)-median(v.0))
            d.median.boot<-lapply(1:anz.boot,median.pb.rfc,
                                  values=values,group.ind=group.ind)
            d.median.boot<- unlist(d.median.boot)
            p.value<- sum(ifelse(abs(d.median.boot)>=abs(d.median.org),1,0))/
                              anz.boot
            return(p.value)
            }
```

Applied to the reduced array we get

```
H.200.RE.median.diff.pb.p.values <-
            esApply(H.200.RE,1, median.pb.p.value.rfc,group.ind=rec.info,
                                                    anz.boot=1000)
```

Again, the bootstrap will be used to calculate 95% confidence intervals for the difference of medians.

```
mean.pb.2g.rfc <-function(i,values,group.ind)
            {
            v.0<-values[group.ind==unique(group.ind)[1]]
            v.1<-values[group.ind==unique(group.ind)[2]]
            v.new.0<- rnorm(length(v.0),mean(v.0),sd(v.0))
            v.new.1<- rnorm(length(v.1),mean(v.1),sd(v.1))
            return(mean(v.new.1)- mean(v.new.0))
            }

median.pb.ci.rfc<- function(values,group.ind,anz.boot=999,alpha=0.05)
            {
            lower<-alpha/2
            upper<-1-alpha/2
            v.0<-values[group.ind==unique(group.ind)[1]]
            v.1<-values[group.ind==unique(group.ind)[2]]
            d.median.org<-(median(v.1)-median(v.0))
            pb.res<-unlist(lapply(1: anz.boot,median.pb.2g.rfc,
                                  values=values, group.ind=group.ind))
            pb.95.ci<-quantile(pb.res,probs=c(lower,upper))
            shift<- d.median.org-median(pb.res)
            pb.95.ci<- pb.95.ci+ shift
            return(c(lower= pb.95.ci[1],upper= pb.95.ci[2]))
            }
```

The 95% CI for the probe sets on the reduced expression set are given by

```
H.200.RE.median.diff.pb.ci <-
            esApply(H.200.RE,1, median.pb.ci.rfc,group.ind=rec.info,
                                        anz.boot=999,alpha=0.05)
```

This calculation took 11 minutes on my laptop. This shows the need for more efficient and faster procedures. The 95% confidence intervals for the fold changes are found by

```
exp(t(H.200.RE.median.diff.pb.ci)).
```

Exrecise 3: Discrimination versus differential gene expression

The scientific objective of a study as this presented by Huang et al. (2003, The Lancet, 361:1590-1596) is to find gene expression as predictors on breast cancer outcome. We will look at the data presented in a more simple way. Does the data help us to find new biomarker which can be used in the follow-up of cancer patients, for example to diagnose recurrence of the tumour?

This exercise follows the arguments given by Pepe et al. (Biometrics, 2003, 59:133-142). They argue as follows: In general, scientists are more interested in identifying genes that are over-expressed, rather than under-expressed, in cancer diagnostic research. This is because detecting the presence of a new aberrant protein in blood is a potentially easier task than is detecting the

reduced level of a normal protein – particularly if that protein is also produced by normal organ tissue in the body of the cancer patient.

There are many genes over-expressed in cancer tissues that cannot lead to screening markers. For example, genes that relate simply to inflammation or growth are not candidates, because those processes also occur naturally in the body.

For the initial selection, the authors propose to include multiple genes that are redundant in the sense that they identify the same cancer samples. So, if one gene proves useless for biomarker development, one can still pursue another that could identify those same cancers.

In the first two exercises, differentially expressed genes were studied. To say that there is differentially expression at gene g was to state that the distribution of gene expression in two groups is different. But what sorts of differences are of particular interest when searching for biomarkers.

This discussion in this exercise concentrates on separation between the distributions of gene expression between two groups.

In the following figure three scenarios are presented which are of different interest but may result in a clear differential expression result in terms of exercise 1 and exercise 2. The distribution of gene expression in the control group (no recurrence) and disease group (recurrence) is presented. The density curve for the disease group is shaded.

```
par(mfrow=c(2,2))
# scenario 1
x<-seq(0,10,0.1)
y.C<-dnorm(x,3,2)
y.D<-dnorm(x,9,0.7)
plot(c(0,10),c(0,max(c(y.C,y.D))),type=”n”,xlab=””,ylab=””,xaxt=”n”,yaxt=”n”)
lines(x, y.C)
lines(x, y.D)
for (i in 1:length(x)) lines(c(x[i],x[i]),c(0,y.D[i]))
title(“Scenario I”)
# scenario 2
x<-seq(0,10,0.1)
y.C<-dnorm(x,3,2)
y.D<-0.4*dnorm(x,9,0.7)+0.6* y.C
plot(c(0,10),c(0,max(c(y.C,y.D))),type=”n”,xlab=””,ylab=””,xaxt=”n”,yaxt=”n”)
lines(x, y.C)
lines(x, y.D)
for (i in 1:length(x)) lines(c(x[i],x[i]),c(0,y.D[i]))
title(“Scenario II”)
# scenario 3
x<-seq(0,10,0.1)
y.C<-dnorm(x,3,2)
y.D<-dnorm(x,5.5,0.7)
plot(c(0,10),c(0,max(c(y.C,y.D))),type=”n”,xlab=””,ylab=””,xaxt=”n”,yaxt=”n”)
lines(x, y.C)
lines(x, y.D)
for (i in 1:length(x)) lines(c(x[i],x[i]),c(0,y.D[i]))
title(“Scenario II”)
par(mfrow=c(1,1))
```

Scenario I:
The ideal situation is represented in scenario I, where there is almost complete separation

between the distributions. In this situation, the expression level is an ideal candidate marker for cancer, because the values are completely different in the control and diseased tissue.

Scenario II:
The marker clearly distinguishes a subset of cancer patients from controls. Looking ahead to population screening, and assuming that gene expression translates roughly into protein expression, scenario II offers with the value 7 a threshold for the test that provides detection of about 41.3% true positives in the diseased and only 2.3% false positives in the control group.

Scenario III:
Scenario III dos not offer a clear possibility of separation.

Using a cutpoint x and declaring all subjects with a measurement above x as diseased introduces a certain false positive rate within the controls FP(x) and a true positive rate within the diseased subjects TP(x). If x moves from the low side of the spectrum of possible values to the high end, the value of FP(x) as well as TP(x) will decrease. Changing the value of x will introduce a curve in a two-dimensional plot given by (FP(x), TP(x)). This curve is called ROC: receiver-operating-characteristic.

The false positive rate FP is also called *1 – specificity*, the true positive rate TP is also called *sensitivity*. The ROC curve for the probe set `34361_at` studied in exercise 1 is given below.

To perform the calculations one has to load the package ROC.

```
require(affy)
require(ROC)
gene.exprs<- exprs(Huang.RE)[4402,]
rec.info<-pData(Huang.RE)$Recurrence
par(pty="s")
plot(rocdemo.sca(rec.info,gene.exprs,dxrule.sca, caseLabel="Recurrence",
                    markerLabel="34361_at"),type="l")
abline(a=0,b=1)
```

Before discussing the graph, some technical remarks have to be given: `par(pty="s")` forces the graph to be square shaped, the area of the graph is equal to 1, `dxrule.sca` is a function which states the diagnostic decision, that a value above the threshold is a diseased case.

The graph shows the ROC curve which is bend towards the upper left corner. A perfect separation is shown by a ROC curve which moves closely to the upper left corner. The separation is insufficient as closer the ROC curve approaches the diagonal. An important measure for the quality of separation is the area under the ROC curve, the AUC. The AUC for the probe set `34361_at` is given by

```
AUC(rocdemo.sca(rec.info,gene.exprs,dxrule.sca, caseLabel="Recurrence",
                      markerLabel="34361_at"))
```

An AUC of 1 corresponds to a ROC curve which is perfectly in the upper left corner. It is of interest to calculate the AUC of the ROC curves for all probe sets of the array.

We apply this idea to the Huang data. To use the `esApply` command, a function is needed with the expression values as first argument. Therefore, the following function will be defined:

```
AUC.overexprs.rfc<-function(gene.exprs,gr.info)
                  {
                  return(AUC(rocdemo.sca(rec.info,gene.exprs,dxrule.sca,
                              caseLabel=" ",markerLabel=" ")))
                  }
```

Applying this function to the reduced Huang expression set gives:

```
H.200.AUC.over.res<-esApply(H.200.RE,1,AUC.overexprs.rfc,gr.info=rec.info)
```

The calculation of 200 AUCs took around 20 seconds. From this result it is of interest to derive a ranking of probe sets with respect to their ability to discriminate. The top ten probe sets are given as

```
ord<-order(H.200.AUC.over.res,decreasing=T)
H.200.AUC.over.res[ord[1:10]]
```

In the next step it is necessary to quantify the degree of confidence in the ranking of a probe set provided by the data. Pepe et al. propose to estimate the probability of a probe set g to be ranked in the top k positions: $P_g(k) = \text{Prob}[\text{Rank}(g) \geq k]$.

The probabilities $P_g(k)$ can be estimated by the bootstrap, with the resampling unit at the tissue level. Thus, when a tissue is included in the bootstrap sample, the entire vector of data relating to all probe sets for that tissue is entered, and genes are ranked within the data set according to the AUC value.

In order to avoid tied data points, the bootstrap will be modified to randomly break ties, by adding small random noise (jitter) to the expression levels. This is done in an effort to make the bootstrap distribution of the rank statistics behind the AUC calculation more reflective of the actual distribution across different realisations of the experiment.

The following code draws the ROC curves for the three scenarios given at the start of exercise 3. Group sizes twice to the Huang example are adopted: 68 controls, 36 patients.

```
par(pty=”s”)
n.con<-68
n.pat<-36
plot(c(0,1),c(0,1),type=”n”,xlab=”1-Specificity”,ylab=”Sensitivity”)
diag.info<-rep(c(0,1),c(n.con,n.pat))
marker.sc.1<-c(rnorm(n.con,3,2),rnorm(n.pat,9,0.7))
# Diseased population in scenario 2 is a mixture
pop.mix<-rbinom(n.pat,1,0.4)
mu.mix<-3+pop.mix*6
sd.mix<-2-pop.mix*1.3
marker.sc.2<-c(rnorm(n.con,3,2),rnorm(n.pat,mu.mix,sd.mix))
marker.sc.3<-c(rnorm(n.con,3,2),rnorm(n.pat,5.5,0.7))
roc.1<- rocdemo.sca(diag.info, marker.sc.1,dxrule.sca,
                  caseLabel=" ",markerLabel=" ")
roc.2<- rocdemo.sca(diag.info, marker.sc.2,dxrule.sca,
                  caseLabel=" ",markerLabel=" ")
roc.3<- rocdemo.sca(diag.info, marker.sc.3,dxrule.sca,
                  caseLabel=" ",markerLabel=" ")
lines(1-roc.1@spec,roc.1@sens,lty=1)
lines(1-roc.2@spec,roc.2@sens,lty=2)
lines(1-roc.3@spec,roc.3@sens,lty=3)
legend(0.6,0.2,paste("Scenario ",1:3,sep=""),lty=1:3)
```

These ideas translate into the following R-code:

```
boot.AUC.rank.prob.rfc<-function(exprset=H.200.RE,k=10,grouping= rec.info,
anz.boot=200)
      {
      mm.org<-exprs(exprset)
      auc.org<- apply(mm.org,1,AUC.overexprs.rfc,gr.info=grouping)
      rank.org<-order(auc.org,decreasing=T)[1:k]
      dd<-dim(mm.org)[[1]]
      ll<-length(grouping)
      ss<-1:ll
      ss.0<-ss[grouping ==unique(grouping)[1]]
      ss.1<-ss[grouping ==unique(grouping)[2]]
      cnt<-0
      res<-rep(0,k)
      while (cnt< anz.boot)
            {
            cnt<-cnt+1
            ss.neu.0<-sample(ss.0,length(ss.0),replace=T)
            ss.neu.1<-sample(ss.1,length(ss.1),replace=T)
            mm.boot<-jitter(mm.org[,c(ss.neu.0,ss.neu.1)])
            auc.boot<- apply(mm.boot,1,AUC.overexprs.rfc,gr.info=grouping)
            rank.boot<-order(auc.boot,decreasing=T)[1:k]
            res<-res+ifelse(rank.org%in%rank.boot,1,0)
            }
      mm.res<- cbind(auc.org[rank.org],res/anz.boot)
      colnames(mm.res)<-c("AUC","Pgk")
      return(mm.res)
      }
```

The calculation of the above procedure will take more than 20 x 200 seconds on my laptop. The computational load is not so big if the ranking is performed for the t-statistic:

```
boot.t.rank.prob.rfc<-function(exprset=Huang.RE,k=40,grouping= rec.info,
anz.boot=10)
      {
      mm.org<-exprs(exprset)
      t.org<- mt.teststat(mm.org,grouping)
      rank.org<-order(t.org,decreasing=T)[1:k]
      dd<-dim(mm.org)[[1]]; ll<-length(grouping)
      ss<-1:ll
      ss.0<-ss[grouping ==unique(grouping)[1]]
      ss.1<-ss[grouping ==unique(grouping)[2]]
      cnt<-0; res<-rep(0,k)
      while (cnt< anz.boot)
            {
            cnt<-cnt+1
            ss.neu.0<-sample(ss.0,length(ss.0),replace=T)
            ss.neu.1<-sample(ss.1,length(ss.1),replace=T)
            mm.boot<-jitter(mm.org[,c(ss.neu.0,ss.neu.1)])
            t.boot<- mt.teststat(mm.boot,grouping)
            rank.boot<-order(t.boot,decreasing=T)[1:k]
            res<-res+ifelse(rank.org%in%rank.boot,1,0)
            }
      mm.res<- cbind(t.org[rank.org],res/anz.boot)
      colnames(mm.res)<-c("t.stat","Pgk")
      return(mm.res)
      }
```

In their article, Pepe et al. discuss also measures for discrimination which are more suited to handle scenario II.