Computer exercises on Experimental Design

Ulrich Mansmann
IMBI, Universität Heidelberg

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())
```

Exercise 1: Sample Size calculation for differential gene expression of independent genes

Setting the parameters

```
alpha<-0.0001             #    Level of individual test
beta<-0.1                 #    Power = 1-beta = 0.9
N.0<-22100                #    Number of not differentially
                          #    expressed probe sets
N.1<-183                  #    Number of differentially
                          #    expressed probe sets
```

Calculation of experiment-wise false positives (FP) and true positives (TP). The simulation looks at 10000 Experiments. The number of TP of each single experiment is a binomial random variable for the N.1 differentially expressed genes with success rate 1-beta = 0.9. The same is true for the number of FP. But because the success rate is quite small (alpha=0.0001) and there are many non-differentially expressed genes (N.0=22100) one also can use the Poisson approximation to this situation.

```
FP<-rpois(10000,N.0*alpha)
TP<-rbinom(10000,N.1,1-beta)
```

Now it is possible to calculate different versions of the FDR.

```
m.FDR<-alpha*N.0/(alpha*N.0+(1-beta)*N.1)
```

The following line calculates the 10000 FDRs for the series of simulated experiments

```
FDR<-FP/(FP+TP)
```

The information on median or mean FDR is given by

```
summary(FDR)
```

The positive FDR is given by

```
p.FDR<-mean(FDR[FP+TP>0])
```

To gain insight in to the conditional FDRs one may study the distribution of positive signals and use the central 95%. Furthermore one may look for these situations on the distribution of the FDR and may calculate the 95% quantile of FDRs over the range of (TP+FP) of interest

```
qq.AP<-quantile(FP+TP,probs=c(0.025,0.975))
c.FDR<-quantile(FDR[FP+TP>qq.AP[1] & FP+TP<qq.AP[2]],0.95)
```

Determine for the given setting by trial and error an alpha value which results in a conditional FDR of 10%.

If alpha and beta are fixed it is possible to determine for a fixed fold change `delta.log.fc` of interest and a fixed standard deviation `sigma` describing the noise present on the chip the number of arrays needed per group to perform the detection of differentially expressed genes. The following function perform the sample size calculation:

```
sample.size.log.FC.two.groups.rfc <-
function (alpha=0.0001, beta=0.1,delta.log.fc=log(2),sigma=0.4)
{
        z.a<-qnorm(1-alpha/2,0,1)
        z.b<-qnorm(1-beta,0,1)
        n<-2*((z.a+z.b)*sigma/delta.log.fc)^2
        return(n)
}
```

Calculate the sample size of the total study for the setting `alpha=0.0001, beta=0.1, delta.log.fc=log(2),sigma=0.4.` The per group result is:

```
> sample.size.log.FC.two.groups.rfc(sigma=0.4)
[1] 17.81723
```

Which changes would result if you can assume that the number of genes with a FC of at least it not 184 but say around 500. What are the implications for the sample size if you require a conditional FDR to 5%.

Exercise 2: Effect of preselection

This exercise analyses the value of the classification procedure proposed by Frederikson et al. (2003) J Cancer Res Clin Oncol 129: 263–271. The performance of the procedure proposed is studied under randomness by creating data without any apriori structure.

Give names to the 25 tissue probes and its numerical counterpart:

```
aa.group.names <-
paste(rep(c("O","A","B","C","D"),rep(5,5)),rep(1:5,5),sep=".")

aa.group.names
 [1] "O.1" "O.2" "O.3" "O.4" "O.5" "A.1" "A.2" "A.3" "A.4" "A.5" "B.1" "B.2"
[13] "B.3" "B.4" "B.5" "C.1" "C.2" "C.3" "C.4" "C.5" "D.1" "D.2" "D.3" "D.4"
[25] "D.5"


aa.gr.exp.25<-rep(1:5,rep(5,5))
```

Generate random data:

```
aa.generate.data.rfc<-
function (anz.genes=5000,gr.names=aa.group.names)
{
        n<-anz.genes
        m<-length(gr.names)
        mat<-matrix(rnorm(n*m,0,1),ncol=m)
        colnames(mat)<-gr.names
        return(mat)
}
```

The following functions present the two selection strategies used by the group of Frederikson. The first is based on a test of difference between the five groups.

```
aa.reduce.f.rfc<-
function (data=aa.data.5000,y.class=aa.gr.exp.25,anz.sel=400)
{
        require(multtest)
        yy<-y.class-1
        nn.1<-length(unique(yy))
        nn.2<-length(yy)
        f.res<-mt.teststat(data,yy,test="f",na=.mt.naNUM,nonpara="n")
        f.pvl<-1-pf(f.res,nn.1-1,nn.2-nn.1)
        f.adj<-mt.rawp2adjp(f.pvl)
        f.sel<-f.adj[[2]][1:anz.sel]
        return(data[f.sel,])
}
```

The second is based on the correlation of a gene with the grouping when the groups get the score 1, 2, 3, 4, 5.

```
aa.reduce.cor.rfc<-
function (data=aa.data.5000,y.class=aa.gr.exp.25,anz.sel=200)
{
        cor.res<-apply(data,1,cor,y=y.class)
        cor.srt<-order(cor.res)
        nn<-length(cor.res)
        ss<-c(1:anz.sel,(nn-anz.sel+1):nn)
        return(data[cor.srt[ss],])
}
```

Now, the classification procedure k-nearest neighbours is performed and the number of correct classifications per group are given as output.

```
aa.generate.class.rfc<-
function (data.train=aa.data.cor,y.class=as.factor(aa.gr.exp.25),k.nn=3)
{
        require(class)
        y.knn<-knn.cv(t(data.train),y.class,k=k.nn)
        return(diag(table(y.knn,y.class)))
}
```

The following function combines for the pre-processing strategy based on the correlation the three steps: 1) generation of unstructured random data, 2.) pre-processing step, 3) classification. The total data consists of 5000 genes of which 200 will be selected.

```
aa.cor.simulation.rfc<-
function (i=1,how.many.genes=5000,how.many.select=200)
{
        dd.full<-aa.generate.data.rfc(anz.genes=how.many.genes)
        dd.redu<-aa.reduce.cor.rfc(data=dd.full,anz.sel=how.many.select)
        return(aa.generate.class.rfc(data.train=dd.redu))
}
```

The following function combines for the pre-processing strategy based on the F-test the three steps: 1) generation of unstructured random data, 2.) pre-processing step, 3) classification. The total data consists of 5000 genes of which 200 will be selected.

```
aa.f.simulation.rfc<-
function (i=1,how.many.genes=5000,how.many.select=200)
{
        dd.full<-aa.generate.data.rfc(anz.genes=how.many.genes)
        dd.redu<-aa.reduce.f.rfc(data=dd.full,anz.sel=how.many.select)
        return(aa.generate.class.rfc(data.train=dd.redu))
}
```

The single parts are now combined in a simulation function which performs `anz.simul` simulations on a scenario with `h.m.g` genes of which `h.m.s` will be selcted by the respective pre-processing procedure.

Correlation based selection:
```
aa.cor.simulation.run.rfc<-
function (anz.simul=10,h.m.g=5000,h.m.s=200)
{
        mm<-matrix(1:anz.simul,ncol=1)
        rr<-apply(mm,1,aa.cor.simulation.rfc,how.many.genes=h.m.g,
                    how.many.select=h.m.s)
        return(rr)
}
```
F-test based selection:
```
aa.f.simulation.run.rfc <-
function (anz.simul=10,h.m.g=5000,h.m.s=200)
{
        mm<-matrix(1:anz.simul,ncol=1)
        rr<-
apply(mm,1,aa.f.simulation.rfc,how.many.genes=h.m.g,how.many.select=h.m.s)
        return(rr)
}
```
The following functions helps to evaluate the simulation results by calculating for each group the simulation runs in which the accuracy of classification was at east as good as a given cut point. Because there are only five items per group the following cut points are of interest: 0,1,2,3,4,5.

```
aa.how.many.above.rfc <-
function (x,cut.point)
{
        return(sum(ifelse(x>=cut.point,1,0)))
}
```
The following results were found in one of my simulation runs.

```
table(apply(aa.cor.simulation.run.1.res,2,
aa.how.many.above.rfc,cut.point=4))
 0  1  2  3  4
 7 38 83 60 12
> table(apply(aa.f.simulation.run.1.res,2,
aa.how.many.above.rfc,cut.point=4))
  3   4   5
  2  31 167
```
Looking at the simulation results, what do you think about the results presented by Frederikson et al. ?

Exercise 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 cut point 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 `399_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.200.RE)[10,]
rec.info<-pData(Huang.200.RE)$Recurrence
par(pty="s")
plot(rocdemo.sca(rec.info,gene.exprs,dxrule.sca, caseLabel="Recurrence",
                    markerLabel= "399_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= "399_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(Huang.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=Huang.200.RE,k=10,grouping=
rec.info, anz.boot=10)
      {
      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.200.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.