TP MISO: tests multiples

Guillemette Marot (guillemette.marot@univ-lille.fr)

1. Dans le cadre de l'analyse de puces à ADN, des milliers de tests sont réalisés simultanément. Il est donc important de corriger les p-values pour les tests multiples. Le jeu de données utilisé dans cette partie est téléchargeable à l'url suivante:

```
microarray.data.url  <- "http://pedagogix-tagc.univ-mrs.fr/courses/ASG1/data/marrays"
expr.url <- file.path(microarray.data.url, "GSE13425_Norm_Whole.txt")
#download.file(url = expr.url, destfile ="./GSE13425_Norm_Whole.txt")
pheno.url<-file.path(microarray.data.url,"phenoData_GSE13425.tab")
#download.file(url = pheno.url,destfile="./phenoData_GSE13425.tab")
expr.matrix<- read.table(expr.url,sep="\t", header = T, row.names = 1)
head(expr.matrix)
pheno.matrix <-  read.table(pheno.url,sep="\t", header = T, row.names = 1)
head(pheno.matrix)
```

La description de ce jeu de données se retrouve grâce aux liens donnés dans la base de données GEO, à partir de l'identifiant GSE13425.

2. Quelles sont les différentes conditions présentes dans cette expérience? On appellera `sample.labels` le vecteur de ces conditions (codé en variable qualitative). Donner la répartition des individus dans ces conditions.

3. Réalisez des t-tests gène à gène pour comparer les différences d'expression entre les conditions "Bh" et "Bo" et récupérez les p-values brutes. On pourra utiliser la fonction `apply` pour accélérer les calculs.

4. Comptez le nombre de gènes différentiellement exprimés lorsqu'on utilise les p-values brutes. Corrigez les p-values pour prendre en compte les tests multiples, soit par la méthode de Bonferroni soit par la méthode de Benjamini et Hochberg (1995), disponibles dans la distribution de base de R via la fonction `p.adjust`. Comptez à nouveau le nombre de gènes différentiellement exprimés après chacune de ces corrections. Commentez.

1 Simulations

Afin de mieux comprendre la nécessité de corriger les p-values brutes pour prendre en compte la multiplicité des tests, on peut simuler un jeu de données dans lequel on connaît la vérité.

1. Simuler une statistique de test qui suit une loi normale centrée réduite sous l'hypothèse nulle et une loi normale de moyenne 2 et de variance 1 sous l'hypothèse alternative. On supposera que le jeu de données contient 10000 variables (e.g. gènes ou protéines), sachant que 500 d'entre elles suivent la loi sous l'hypothèse alternative (e.g. gènes ou protéines différentiellement exprimés).

```
set.seed(123)
ngenes = 10000
nDE = 500
simulstat <- rnorm(ngenes,mean=0,sd=1)
indDE <- sample(1:ngenes,nDE)
simulstat[indDE] <- rnorm(nDE,mean=2,sd=1)
```

```
plot(density(simulstat),main="simulated data")
points(sort(simulstat[-indDE]),(1-nDE/ngenes)*dnorm(sort(simulstat[-indDE]),mean=0,sd=1),
       type="l",col=2)
points(sort(simulstat[indDE]),nDE/ngenes*dnorm(sort(simulstat[indDE]),mean=2,sd=1),
       type="l",col=3)
legend("topright",c("simulstat", "N(0,1)", "N(2,1)"), col=1:3, lty=1)
```

2. Calculer les p-values brutes et compter le nombre de vrais positifs et faux positifs en choisissant un seuil de 5% pour le risque de première espèce pour chaque test.

3. Ajuster les p-values brutes pour contrôler le taux de faux positifs (FDR - False Discovery Rate) avec la méthode de Benjamini et Hochberg (1995). Compter le nouveau nombre de faux positifs.