1: Rapport d'analyse des données Agilent

Jerome Ambroise

2 mars 2017

Dans le document, ci dessous, nous allons analyser des données microarray générées avec une plateforme Agilent.

Les données proviennent de l'étude "MicroArray Quality Control (MAQC) project" dont un des papier vous est fourni. Dans ce document, nous analysons 5 échantillons A (A1, A2, A3, A4 et A5) et 5 échantillons B (B1, B2, B3, B4 et B5) et l'objectif est d'identifier les gènes qui sont modulés.

1: Chargement des packages et importation des données.

Nous commençons par charger le package Limma. Limma est un package bioconductor qui dispose d'un "user guide" très complet

```
rm(list=ls())
library(limma)
```

```
## Warning: package 'limma' was built under R version 3.2.4
```

Nous importons les données qui sont situées dans le répertoire 1-data/AGL. Il s'agit de 10 fichiers '.txt'.

```
files <- list.files('1-data/AGL/')
files <- paste0('1-data/AGL/',files)
agl <- read.maimages(files, source="agilent", green.only=TRUE)
```

```
## Read 1-data/AGL/AG1_1_A1.txt
## Read 1-data/AGL/AG1_1_A2.txt
## Read 1-data/AGL/AG1_1_A3.txt
## Read 1-data/AGL/AG1_1_A4.txt
## Read 1-data/AGL/AG1_1_A5.txt
## Read 1-data/AGL/AG1_1_B1.txt
## Read 1-data/AGL/AG1_1_B2.txt
## Read 1-data/AGL/AG1_1_B3.txt
## Read 1-data/AGL/AG1_1_B4.txt
## Read 1-data/AGL/AG1_1_B5.txt
```

2: Pré-traitement des données.

Les données de microarray sont généralement pré-traitées en 3 étapes:

1: la correction du background

2: la normalisation

3: la transformation (généralement log2)

Nous commençons par effecture la correction du background

```
agl_BC <- backgroundCorrect(agl,method='subtract')
```

Ensuite, nous normalisons les données. Durant cette normalisation, vous pouvez constater que R applique également une transformation logarithmique (log2)

```
agl_BC_NO <- normalizeBetweenArrays(agl_BC, method="quantile")
```

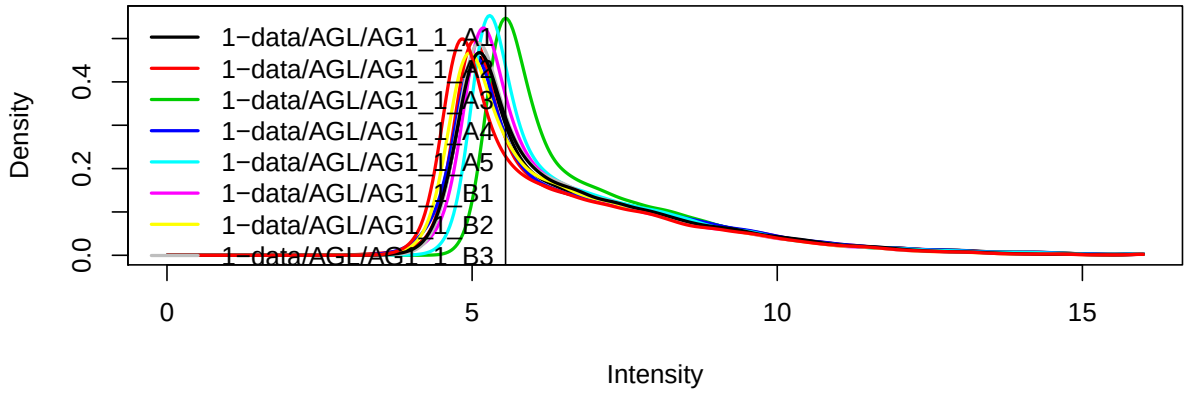
```
## Warning in normalizeBetweenArrays(agl_BC, method = "quantile"): production
## de NaN
```

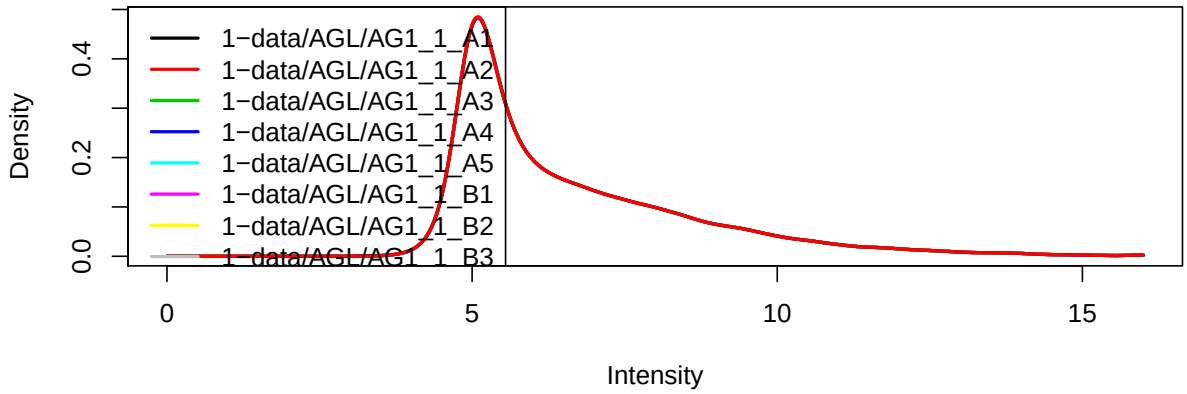
Nous pouvons afficher l'effet de la normalisation dans des graphiques de densité.

```
par(mfrow=c(2,1))
plotDensities(agl_BC,from=0,to=16)
```

```
## Warning in plotDensities.EListRaw(agl_BC, from = 0, to = 16): production de
## NaN
```

```
plotDensities(agl_BC_NO,from=0,to=16)
```





Nous pouvons également créer un fichier pdf avec ces graphiques.

```
pdf('2-result/1a-agilent-normalisation.pdf',width=10,height=10)
par(mfrow=c(2,1))
plotDensities(agl_BC,from=0,to=16)
```

```
## Warning in plotDensities.EListRaw(agl_BC, from = 0, to = 16): production de
## NaN
```

```
plotDensities(agl_BC_NO,from=0,to=16)
dev.off()
```

```
## pdf
##   2
```

3: Création des modèles statistiques pour détecter les gènes modulés

Pour rappel, le but est de comparer les niveaux d'expression de chacun des gènes entre les 5 échantillons 'A' et les 5 échantillons 'B'. Nous commencçons par spécifier à R le design de notre expérience

```
mycondition <- c(rep('A',5),rep('B',5))
mydesign <- model.matrix(~mycondition )
print(mydesign)
```

```
##    (Intercept) myconditionB
## 1            1            0
## 2            1            0
## 3            1            0
## 4            1            0
## 5            1            0
## 6            1            1
## 7            1            1
## 8            1            1
## 9            1            1
## 10           1            1
## attr(,"assign")
## [1] 0 1
## attr(,"contrasts")
## attr(,"contrasts")$mycondition
## [1] "contr.treatment"
```

Une fois que le modèle est spécifié, il est possible de créer les modèles pour chacun des gènes et de calculer les statistiques de test et les p-valeurs associées à chacun des gènes. Cela se fait en deux étapes avec les fonctions 'lmFit' et 'eBayes' du package 'limma'.

```
fit <- lmFit(agl_BC_NO, design=mydesign)
efit <- eBayes(fit)
```

Une fois les modèles construits, on peut aller chercher les informations qui nous intéressent et les regrouper dans une data.frame. Il s'agit de 3 éléments

1: Le coefficient d'intérêt du modèle construit pour chacun des gènes. Ce coefficient correspond à la différence moyenne entre le niveau d'expression des échantillons A et B, dans l'échelle log2. Il s'agit donc d'un 'log2 fold-change'.

2: la p-valeur d'intérêt du modèle construit pour chacun des gènes.

3: l'annotation (çad le nom) de chacun des gènes

```
coefficient.agl <- fit$coefficients
head(coefficient.agl)
```

```
##   (Intercept) myconditionB
## 1    4.815673 -0.211460021
## 2    4.721416 -0.009336099
## 3    5.933125  0.186056111
## 4    6.104002  2.025768732
## 5    7.094341  0.027706983
## 6    5.285616 -0.485834199
```

```
coefficient.agl <- coefficient.agl[,2]
pvalue.agl <- efit$p.value
head(pvalue.agl)
```

```
##    (Intercept) myconditionB
## 1 1.629338e-15 1.063359e-01
## 2 2.673364e-14 9.517088e-01
## 3 1.599642e-18 4.280556e-02
## 4 6.730785e-19 1.238518e-11
## 5 8.240053e-19 7.695221e-01
## 6 3.046974e-16 1.242698e-03
```

```
pvalue.agl <- pvalue.agl[,2]
annotation <- efit$genes
annotation <- annotation['GeneName']
resultat.agl <- data.frame(annotation,coefficient.agl,pvalue.agl)
head(resultat.agl,n=10)
```

```
##          annotation coefficient.agl   pvalue.agl
## 1     BrightCorner    -0.211460021 1.063359e-01
## 2  NegativeControl    -0.009336099 9.517088e-01
## 3          L3MBTL2     0.186056111 4.280556e-02
## 4            APBA2     2.025768732 1.238518e-11
## 5           MAP3K6     0.027706983 7.695221e-01
## 6           ZNF121    -0.485834199 1.242698e-03
## 7           Pro25G    -0.338956516 1.140560e-02
## 8              RET    -0.145093094 1.407250e-01
## 9         BX094364    -0.128282301 2.037069e-01
## 10 ENST00000333722    -0.337689249 6.869596e-02
```

4: Classement des résultats et ajustement des p-valeurs

Nous allons maintenant classer les gènes selon les p-valeurs. Nous allons également calculer les p-valeurs ajustées selon la méthode de 'Benjamini-Hochberg'.

```
resultat.agl <- resultat.agl[sort.list(resultat.agl$pvalue.agl),]
adj.pvalue.agl <- p.adjust(resultat.agl$pvalue.agl,method='BH')
resultat.agl <- data.frame(resultat.agl,adj.pvalue.agl)
head(resultat.agl)
```

```
##       annotation coefficient.agl   pvalue.agl adj.pvalue.agl
## 13379      STMN4        7.659475 2.339391e-21   1.027718e-16
## 26689       AHSG       -9.362454 5.764477e-21   1.266196e-16
## 24728       HBG1       -9.648858 1.514123e-20   2.217232e-16
## 24798    SPARCL1        8.964683 3.257294e-20   2.601207e-16
## 20678      STMN2        9.439867 4.136058e-20   2.601207e-16
## 8188       SCN2B        6.885920 4.433064e-20   2.601207e-16
```

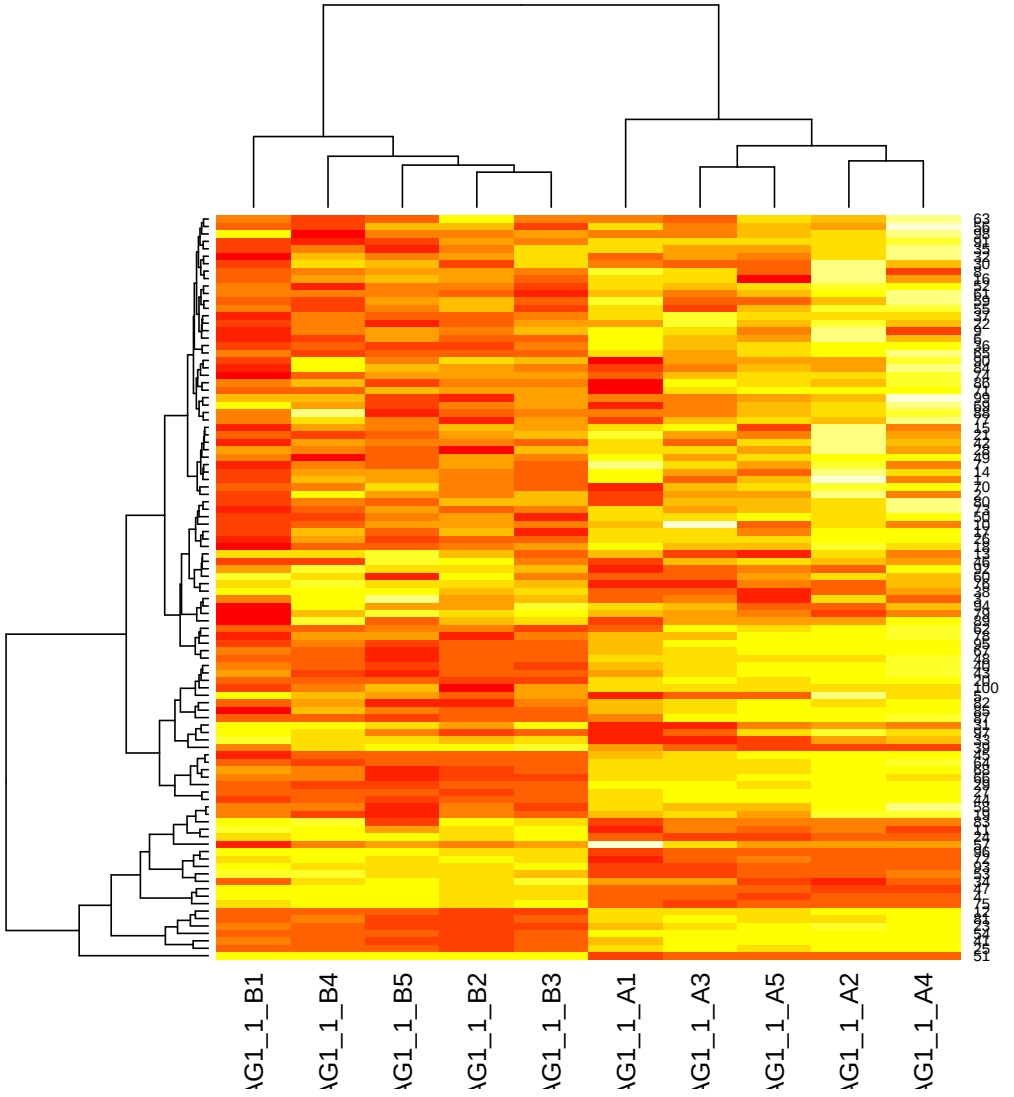
Finalement, nous écrivons tous les résutlats dans un fichier '.txt' dans le répertoire '2-result'.

```
write.table(resultat.agl,'2-result/1-fold-change-agl.txt',sep='\t',row.names=F)
```

4: Quelques graphiques

Nous allons représenter une heatmap des 100 premiers gènes (choix arbitraire des gènes).

```
expression <- agl_BC_NO$E[1:100,]
heatmap(expression)
```



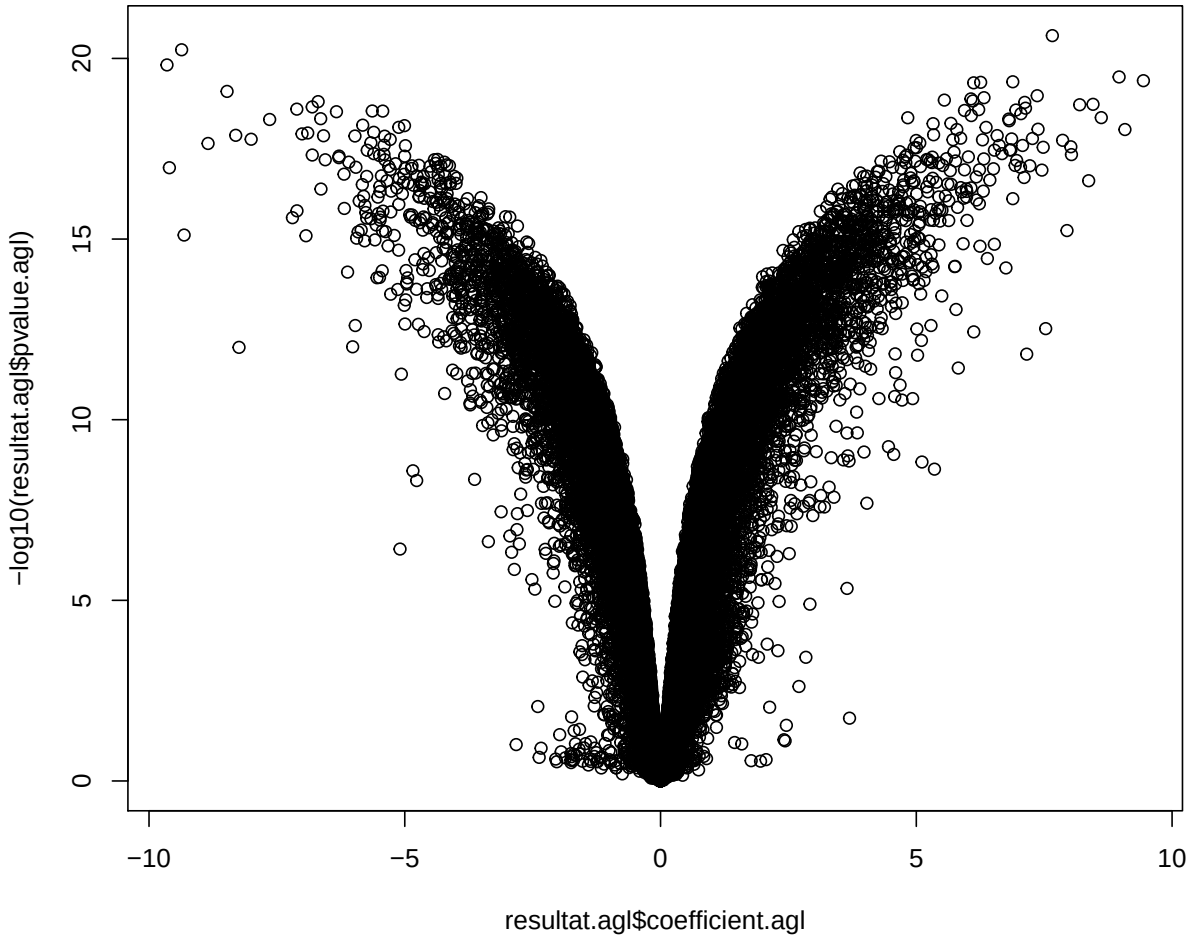
Nous pouvons afficher ce graphique dans une fichier pdf.

```
pdf('2-result/1b-agilent-heatmap.pdf',width=8,height=8)
expression <- agl_BC_NO$E[1:100,]
heatmap(expression)
dev.off()
```

```
## pdf
##   2
```

Nous allons représenter un volcano-plot.

```
plot(resultat.agl$coefficient.agl,-log10(resultat.agl$pvalue.agl))
```



Nous pouvons afficher ce graphique dans une fichier pdf.

```
pdf('2-result/1c-agilent-heatmap.pdf',width=8,height=8)
plot(resultat.agl$coefficient.agl,-log10(resultat.agl$pvalue.agl))
dev.off()
```

```
## pdf
##   2
```