TP3 Analyse differentielle

Justine Guégan - j.guegan-ihu@icm-institute.org
Guillaume Meurice - guillaume.meurice@gustaveroussy.fr

08 décembre 2016

Les données nécessaires à cette séance se trouvent sur le site web :

https://pf-bb.github.io/CentraleSupelec-R-genomics/TP3.html.

Objectif

L'objectif du TP est d'étudier la modulation de l'expression des gènes entre des échantillons de cancer du sein et des échantillons non tumoraux. Pour cela, des expériences de RNA-seq ont été réalisées. Il s'agit ici de mener une analyse différentielle des données de séquencage. Cette analyse différentielle permettra de mettre en évidence quels gènes sont différentiellement exprimés entre les différents sous-type de cancer du sein, et du tissu normal.

Répondez aux questions dans un document Rmarkdown produisant un fichier **PDF** ou **HTML**.

Données

Les ARNm de 17 échantillons de tumeurs du sein de 3 types, HER2 positif (HER2), triple négatif (TNBC), non triple négatif (Non-TNBC), et 3 échantillons de sein normal (NBS) ont été séquencés par Illumina HiSeq2000.

Analyse des données

Lors du TP précedent, vous aviez utilisé une matrice de comptage brute, que vous aviez nettoyé afin de supprimer les gènes qui ne s'expriment dans aucun échantillons. Nous vous proposons de reprendre les analyses à ce stade.

1. Lecture des fichiers de données

> **Question 1.1** : Chargez en mémoire la table de comptage créée lors du TP2 (fichier `counts_normalized.txt`). Quelle est la classe de l'objet créé ? Transformez le en matrice. Chargez en mémoire le fichier de description des échantillons (fichier `annot_sample.txt`).

```
count = read.delim("data_TP3/counts_normalized.txt", sep="\t", row.names = 1)
count = as.matrix(count)
annot = read.delim("data_TP3/annot_sample.txt")
```

2. Création des `MA-plot`

Le MA-plot est une figure permettant de représenter présentant de façon synthètique une comparaison de 2 groupes d'intérêt. Nous vous proposons ici quelques définitions permettant de comprendre comment est construit un MA-plot, en définissant d'abord le **Fold-Change** (FC), puis les valeurs **M** et **A**.

Soit $\bar{x}_1$ et $\bar{x}_2$, définissant respectivement pour un gène j donné, la moyenne des valeurs d'expression pour le groupe 1 et le groupe 2.

- Le FC se défini comme le ratio de la moyenne des valeurs d'expressions entre deux groupes d'intérêt. Il se calcule sans la transformation logarithmique. Si on note, pour un gène j donné, $\bar{x}_1$ la moyenne du groupe 1 et $\bar{x}_2$ la moyenne du groupe 2, alors le **FC** du gène j vaut :

$$FC_j = \frac{\bar{x}_1}{\bar{x}_2}.$$

  On préfère, pour des raisons pratiques, raisonner sur le *log Fold-Change* (logFC), qui est simplement obtenu en prenant le logarithme naturel du FC. Ainsi :

  - si $log(FC_j) > \kappa$, on dit que le gène j est sur-exprimé dans le groupe 1 par rapport au groupe 2,
  - si $log(FC_j) < -\kappa$, on dit que le gène j est sous-exprimé dans le groupe 1 par rapport au groupe 2,

  avec κ une certaine valeur seuil[1], qui dépend beaucoup de l'expérience. Habituellement, on choisit $\kappa = 1$ car cela signifie que la valeur moyenne d'expression est deux fois plus (resp. moins) grande dans un groupe que dans l'autre.

- La valeur **M** correspond simplement au log Fold-Change (logFC) et est donc définie comme suit :

$$M_j = log2(FC) = log2(\frac{\bar{x}_1}{\bar{x}_2})$$

- La valeur **A** correspond à la moyenne des log2 des moyennes des valeurs d'expression :

$$A_j = \frac{1}{2} * [log2(\bar{x}_1) + log2(\bar{x}_2)]$$

Question 2.1 : La fonction computeMean.

Créez une fonction `computeMean` permettant de calculer, pour tous les gènes, la moyenne des valeurs d'expression pour un groupe d'échantillons donné. Cette fonction prend en entré les paramètres suivants :

- `condition` : le vecteur de description de la `condition` des échantillons
- `count` : une matrice de comptage
- `label.grp` : le label du groupe d'intérêt.

Cette fonction retourne un vecteur contenant les moyennes d'expressions pour l'ensemble des gènes de la matrice de comtage.

```
computeMean = function( condition, count, label.grp){
  idx = which( condition == label.grp)
  if (length(idx) > 0){
    m = apply(count[,c(idx)], 1,mean)
    return(m)
  }
  else{
    warning(paste("Le label '",label.grp, "' n'est pas contenu dans le vecteur d'annotation"))
    return(NULL)
  }
}
```

[1] Attention, cette notation de κ pour un seuil de logFC n'est pas universelle, elle est même spécifique de cet énoncé TP. Si vous choisissez d'appeler le seuil ainsi dans votre rapport, n'oubliez pas de le préciser !

Question 2.2 : MA-plots

Afficher les MA-plots pour les groupes suivants :

- HER2 versus NBS
- TNBC versus NBS
- Non-TNBC versus NBS

Pour chaque graphique, ajouter une ligne rouge à y = 0, et deux lignes bleu, respectivement à y = -1, et y = 1. Afficher le titre du graphique, ainsi que le nom des axes. Interpretez ces figures. Que representent les gènes situés au dessus de la ligne y = 1 ? Que representent les gènes situés au dessous de la ligne y = -1 ?

```
m_nbs   = computeMean(annot$condition, count, "NBS")
m_her2  = computeMean(annot$condition, count, "HER2")
m_tnbc  = computeMean(annot$condition, count, "TNBC")
m_ntnbc = computeMean(annot$condition, count, "Non-TNBC")

M_HER2_vs_NBS = log2(m_her2 / m_nbs)
A_HER2_vs_NBS  = 0.5 * ( log2(m_her2) + log2(m_nbs) )
plot(A_HER2_vs_NBS, M_HER2_vs_NBS, pch = 16, cex = 0.6, xlab = "A", ylab = "M", main = "HER2 vs NBS")
abline(h=c(-1,0,1), col = c("blue", "red", "blue"), lwd = 2, lty = 2)

M_TNBC_vs_NBS = log2(m_tnbc / m_nbs)
A_TNBC_vs_NBS  = 0.5 * ( log2(m_tnbc) + log2(m_nbs) )
plot(A_TNBC_vs_NBS, M_TNBC_vs_NBS, pch = 16, cex = 0.6, xlab = "A", ylab = "M", main = "TNBC vs NBS")
abline(h=c(-1,0,1), col = c("blue", "red", "blue"), lwd = 2, lty = 2)

M_NTNBC_vs_NBS = log2(m_ntnbc / m_nbs)
A_NTNBC_vs_NBS  = 0.5 * ( log2(m_ntnbc) + log2(m_nbs) )
plot(A_NTNBC_vs_NBS, M_NTNBC_vs_NBS, pch = 16, cex = 0.6, xlab = "A", ylab = "M", main = "Non-TNBC vs N
abline(h=c(-1,0,1), col = c("blue", "red", "blue"), lwd = 2, lty = 2)
```

3. Analyse différentielle

Un gène est déclaré différentiellement exprimé si une différence observée ou un changement d'expression entre deux conditions expérimentales est significativement statistique, c'est-à-dire plus grande que la valeur attendue.

Nous avons précedemment calculé les logFC, il faut donc calculer les p-values associées à ces logFC.

Lorsque l'on fait un test d'hypothèses, une manière synthétique de représenter le résultat du test est la p-value (moins couramment appelée valeur p). Par définition, la p-value obtenue représente la probabilité sous hypothèse nulle d'obtenir une statistique encore plus atypique que celle obtenue à la suite de notre expérience.

A partir du calcul de la p-value, la conclusion d'un test d'hypothèses se déroule comme suit :

- si la p-value est en dessous du seuil de rejet que j'ai choisi (habituellement : 0.05), je rejette l'hypothèse nulle,
- sinon, je ne peux pas rejeter l'hypothèse nulle.

Question 3.1 : Transformez la matrice `count` en `countLog2`. Calculez, pour tous les gènes (ie les lignes de `countLog2`), les p-values d'un test de Student comparant les deux moyennes des groupes HER2 et NBS en utilisant la fonction t.test et ses paramètres par défaut. Combien de ces p-values sont-elles en dessous du seuil classique de 0.05 ?

```r
countLog2 = log2(count)
idxHER2 = which(annot$condition == "HER2")
idxNBS  = which(annot$condition == "NBS")

pv = c()
for (i in 1:nrow(countLog2)){
  tt = t.test( countLog2[i,idxHER2], countLog2[i,idxNBS])
  pv = c(pv, tt$p.value)
}

length(which(pv <= 0.05))
```

Question 3.2 : Correction pour les test multiples. Les tests d'hypothèses n'ont pas été créés dans l'optique d'être utilisés plus de 20 000 fois de façon successive : si on suit la procédure habituelle, on risque de rejeter l'hypothèse nulle à tort beaucoup trop souvent. La conséquence immédiate et néfaste de ces tests multiples est d'augmenter artificiellement le nombre de gènes différentiellement exprimés. Une correction pour les tests multiples est donc nécessaire. La procédure la plus simple est de diminuer le seuil de rejet (c'est la procédure dite de Bonferroni). Nous allons utiliser dans ce TP la procédure qui est utilisée classiquement en transcriptomique : la procédure de Benjamini-Hochberg. Utilisez sur le vecteur des p-values calculées précédemment la procédure `p.adjust` en attribuant à l'argument method la valeur "BH". Après correction, combien de p-values ajustées se trouvent en dessous du seuil de 0.05 ?

```r
p.adj = p.adjust(pv, method = "BH")
length(which(p.adj <= 0.05))
```

Question 3.3 : Répétez les questions 3.1 et 3.2 pour les contrastes "TNBC vs NBS", et "Non-TNBC vs NBS"

```r
countLog2 = log2(count)
idxTNBC = which(annot$condition == "TNBC")
idxNBS  = which(annot$condition == "NBS")

pv = c()
for (i in 1:nrow(countLog2)){
  tt = t.test( countLog2[i,idxTNBC], countLog2[i,idxNBS])
  pv = c(pv, tt$p.value)
}

length(which(pv <= 0.05))

p.adj = p.adjust(pv, method = "BH")
length(which(p.adj <= 0.05))
```

```r
countLog2 = log2(count)
idxNTNBC = which(annot$condition == "Non-TNBC")
idxNBS  = which(annot$condition == "NBS")

pv = c()
for (i in 1:nrow(countLog2)){
  tt = t.test( countLog2[i,idxNTNBC], countLog2[i,idxNBS])
  pv = c(pv, tt$p.value)
```

```
}

length(which(pv <= 0.05))
p.adj = p.adjust(pv, method = "BH")
length(which(p.adj <= 0.05))
```

4. Représentations graphiques et conclusions

Une liste de gènes différentiellement exprimés est caractérisée par deux seuils :

- un seuil sur le log Fold-Change,
- un seuil sur la p-value corrigée

Question 4.1 : Combien de sondes passent un seuil en logFC de 1 et un seuil sur la p-value ajustée de 0.05 pour les 3 contrastes suivants :

- TNBC versus NBS
- Non-TNBC versus NBS
- HER2 vesus NBS

```
countLog2 = log2(count)

m_nbs   = computeMean(annot$condition, count, "NBS")
m_her2  = computeMean(annot$condition, count, "HER2")
m_tnbc  = computeMean(annot$condition, count, "TNBC")
m_ntnbc = computeMean(annot$condition, count, "Non-TNBC")

M_HER2_vs_NBS = log2(m_her2 / m_nbs)
M_TNBC_vs_NBS = log2(m_tnbc / m_nbs)
M_NTNBC_vs_NBS = log2(m_ntnbc / m_nbs)

idxNTNBC = which(annot$condition == "Non-TNBC")
idxNBS  = which(annot$condition == "NBS")
pv_NTNBC_NBS = c()
for (i in 1:nrow(countLog2)){
  tt = t.test( countLog2[i,idxNTNBC], countLog2[i,idxNBS])
  pv_NTNBC_NBS = c(pv_NTNBC_NBS, tt$p.value)
}

p.adj_NTNBC_NBS = p.adjust(pv_NTNBC_NBS, method = "BH")

idxTNBC = which(annot$condition == "TNBC")
idxNBS  = which(annot$condition == "NBS")
pv_TNBC_NBS = c()
for (i in 1:nrow(countLog2)){
  tt = t.test( countLog2[i,idxTNBC], countLog2[i,idxNBS])
  pv_TNBC_NBS = c(pv_TNBC_NBS, tt$p.value)
}
p.adj_TNBC_NBS = p.adjust(pv_TNBC_NBS, method = "BH")

idxHER2 = which(annot$condition == "HER2")
idxNBS  = which(annot$condition == "NBS")
pv_HER2_NBS = c()
```

```r
for (i in 1:nrow(countLog2)){
  tt = t.test( countLog2[i,idxHER2], countLog2[i,idxNBS])
  pv_HER2_NBS = c(pv_HER2_NBS, tt$p.value)
}
p.adj_HER2_NBS = p.adjust(pv_HER2_NBS, method = "BH")

length(intersect( which(p.adj_HER2_NBS  <= 0.05), which(abs(M_HER2_vs_NBS)  >= 1) ))
length(intersect( which(p.adj_TNBC_NBS  <= 0.05), which(abs(M_TNBC_vs_NBS)  >= 1) ))
length(intersect( which(p.adj_NTNBC_NBS <= 0.05), which(abs(M_NTNBC_vs_NBS) >= 1) ))
```

Question 4.2 : Une première représentation graphique permettant de synthétiser ce résultat est une représentation dite en volcan : il s'agit de représenter, pour toutes les sondes, un graphe bivarié, avec en abscisse le logFC et en ordonnée -log10(p-value ajustée) (ou -log10(p.value)). Faites une représentation en volcan de votre analyse différentielle (avec la fonction plot). Représentez sur ce graphe les seuils sur le logFC et la p-value ajustée (avec la fonction abline). Représentez de deux couleurs différentes les sondes sur- et sous-exprimées (avec l'argument col de la fonction plot).

```r
par(mfrow = c(1,2))
plot(M_HER2_vs_NBS, -log10(pv_HER2_NBS), main = "HER2 vs NBS", pch = 16, cex = 0.7, xlab = "M", ylab = "
idxUp = intersect( which(pv_HER2_NBS  <= 0.05), which(M_HER2_vs_NBS  >=  1 ))
idxDn = intersect( which(pv_HER2_NBS  <= 0.05), which(M_HER2_vs_NBS  <= -1 ))
points(M_HER2_vs_NBS[idxUp], -log10(pv_HER2_NBS[idxUp]), col = "red"  , pch = 16, cex = 0.7)
points(M_HER2_vs_NBS[idxDn], -log10(pv_HER2_NBS[idxDn]), col = "green", pch = 16, cex = 0.7)
abline (h = -log10(0.05),  lty = 2, lwd = 2, col = "blue")
abline (v = c(-1,1), lty = 2, lwd = 2, col = "blue")

plot(M_HER2_vs_NBS, -log10(p.adj_HER2_NBS), main = "HER2 vs NBS", pch = 16, cex = 0.7, xlab = "M", ylab
idxUp = intersect( which(p.adj_HER2_NBS  <= 0.05), which(M_HER2_vs_NBS  >=  1 ))
idxDn = intersect( which(p.adj_HER2_NBS  <= 0.05), which(M_HER2_vs_NBS  <= -1 ))
points(M_HER2_vs_NBS[idxUp], -log10(p.adj_HER2_NBS[idxUp]), col = "red"  , pch = 16, cex = 0.7)
points(M_HER2_vs_NBS[idxDn], -log10(p.adj_HER2_NBS[idxDn]), col = "green", pch = 16, cex = 0.7)
abline (h = -log10(0.05),  lty = 2, lwd = 2, col = "blue")
abline (v = c(-1,1), lty = 2, lwd = 2, col = "blue")
```

Question 4.3 : Enfin, représentez à l'aide de la fonction pheatmap une représentation des mesures d'expression (contenue dans la matrice X) uniquement pour les transcrits différentiellement exprimés, et dont la légende contient les informations contenues dans le fichier d'annotations. Utilisez l'option `scale='row'` . Interprétez la figure.

```r
library(pheatmap)
idxHER2 = which(annot$condition == "HER2")
idxNBS  = which(annot$condition == "NBS")
idx = intersect( which(p.adj_HER2_NBS  <= 0.05), which(abs(M_HER2_vs_NBS)  >= 1) )
countRed = count[idx,c(idxHER2, idxNBS)]
pheatmap(countRed, show_colnames = TRUE, show_rownames = FALSE, scale='row')

idxTNBC = which(annot$condition == "TNBC")
idxNBS  = which(annot$condition == "NBS")
idx = intersect( which(p.adj_TNBC_NBS  <= 0.05), which(abs(M_TNBC_vs_NBS)  >= 1) )
countRed = count[idx,c(idxTNBC, idxNBS)]
pheatmap(countRed, show_colnames = TRUE, show_rownames = FALSE, scale='row')
```

```r
idxNTNBC = which(annot$condition == "Non-TNBC")
idxNBS  = which(annot$condition == "NBS")
idx = intersect( which(p.adj_NTNBC_NBS  <= 0.05), which(abs(M_NTNBC_vs_NBS)  >= 1) )
countRed = count[idx,c(idxNTNBC, idxNBS)]
pheatmap(countRed, show_colnames = TRUE, show_rownames = FALSE, scale='row')
```