TP de la séance 4, Clustering

Diplôme Interuniversitaire en Bioinformatique intégrative (DU-Bii 2019)

Anne Badel, Frederic Guyon & Jacques van Helden

2019-02-20

Contents

Introduction

But de ce TP

Le tutoriel ci-dessous vous guidera pas-à-pas dans l'utilisation de fonctions **R** pour effectuer un clustering sur des profils transcriptomiques RNA-seq.

Source des données

Les données sont issues de la base Recount2 (https://jhubiostatistics.shinyapps.io/recount/). Nous avons sélectionné l'étude **TCGA** (The Cancer Genome Atlas; https://cancergenome.nih.gov/), regroupant des données RNA-seq pour plus de 12.000 patients souffrant de différents types de cancer. Nous nous intéressons ici uniquement aux données **Breast Invasive Cancer** (**BIC**) concernant le cancer du sein.

Les données ont été préparées pour vous, selon la procédure détaillée au cours sur l'analyse différentielle de données RNA-seq.

1. Filtrage des gènes à variance nulle et de ceux ccontenant trop de zéros.
2. Normalisation (méthode robuste aux outliers)
3. Analyse différentielle multi-groupes (en utilisant le package Bioconductor `edgeR`).
4. Correction des P-valeurs nomiables pour tenir compte des tests multiples (nous avons testé ici ~20.000 gènes). Nous estimons le le False Discovery Rate (FDR) selon la méthode de Benjamini-Hochberg (fonction R `p.adjust(all.pvalues, method="fdr")`).
5. Sélection de gènes différentiellement exprimés sur base d'un seuil $\alpha = 0.05$ appliqué au FDR.

Choisir son environnement de travail

Vous pouvez choisir de travailler soir sur le cluster core de l'IFB soit sur les ordinateurs de Paris-Diderot.

1. Sur le **cluster de l'IFB**
    - ouvrez une connexion au serveur RStudio https://rstudio.cluster.france-bioinformatique.fr/ et identifiez-vous
2. Sur les machines de la **salle d'ordinateurs de Paris-Diderot**
    - vous devez avoir la commande suivante dans votre `.bashrc` :

```
source /opt/sdv/anaconda/etc/profile.d/conda.sh`
```

- puis vous devez **lancer l'environnement conda** adéquat :

```
conda activate
```

- et enfin lancer le serveur Rstudio au moyen de la commande bash: `rstudio`

Dossier partagé contenant les données

Les données sont dans un répertoire partagé, dont le chemin dépend du serveur auquel vous êtes connectézs. Nous allons définir une variable `data.folder` qui indiquera le chemin de ce dossier partagé, en fonction du serveur.

1. Sur le serveur Rstudio de l'**IFB-core-cluster**, les données sont dans le répertoire `/shared/projects/du_bii_2019/data`

```
## Liste des fichiers de données
data.folder <- "/shared/projects/du_bii_2019/data/module3/seance4/BIC/"
```

2. Sur les machines de Paris-Diderot, elles sont dans le répertoire `/home/sdv/dubii/data-m3/s4`

```
## Liste des fichiers de données
data.folder <- "/home/sdv/dubii/data-m3/s4/BIC/"
```

Contenu du dossier de données

Utilisez les commandes R suivantes:

- `list.files()` pour vérifier le contenu du dosser `data.folder`,
- `file.size()` pour calculer la taille de ces fichiers.

Astuces:

- `list.files()` retourne par défaut le nom de fichier, mais avec l'option `full.names=TRUE` vous obtiendrez le chemin complet.
- Calculez la taille des fichiers en bytes et en Megabytes ($1Mb = 1024 \cot 1024 \cdot b$), sachant que pour chaque conversion il faut diviser par 1024.
- Vous pouvez consulter notre solution à l'aide du code suivant (cliquer sur ***Code*** pour l'afficher).

```
list.files(path = data.folder) # List the data files
file.size(list.files(path = data.folder, full.names = TRUE)) # Get the size of the data files, in bytes

file.size(list.files(path = data.folder, full.names = TRUE))/ 1024^2 # Get the size of the data files In
```

Lire le tableau de valeurs d'expression

Nous allons maintenant lire le fichier d'expression. Pour cela, nous concaténons le chemin du dossier de données et le nom du fichier d'expressiion (`BIC_diff_exp.tsv`). Ce fichier contient le comptages de lectures RNA-seq par gène, avec une sélection des gènes déclarés positifs pour letest de comparaison de moyennes multiples (voir-ci-dessus).

```
# Define the path of the expression file
BIC.expr.file <- file.path(data.folder, "BIC_log2-norm-counts_edgeR_DEG_fdr_0.01.tsv.gz")

# Load expression
BIC.expr <- read.table(file = BIC.expr.file, h=TRUE)
```

Mesure de la taille des données

Prenez le temps d'identifier

- la taille du jeu de données
- le nombre d'individus
- le nombre de variables

Remarque : Classiquement, en analyse de données, les individus sont les lignes du tableau de données, les colonnes sont les variables.

Pour des raisons historiques, en analyse transcriptomique les données sont toujours fournies avec

- 1 ligne = 1 gène
- 1 échantillon biologique = 1 colonne

Cette convention a été établie en 1997, lors des toutes premières publications sur le transcriptome de la levure. Dans ces études, l'objet d'intérêt (l'"éindividu") était le gène, et les variables étaient ses mesures d'expression dans les différentes conditions testées.

Pour l'anlayse de tissus cancéreux, on considère au contraire que l'"objet" d'intérêt est l'échantillon prélevé sur le patient, et les variables sont les mesures d'expression des différents gènes chez un patient.

Ce qui implique de faire attention, et éventuellement de travailler sur la matrice transposée (fonction `t` en R) pour utiliser correctement les fonctions classiques.

```
## Les données d'expression
dim(BIC.expr)

## Les noms des lignes correspondent à des gènes
head(rownames(BIC.expr))

## Les noms des colonnes correspondent à des échantillons
head(colnames(BIC.expr))

## transposons la table d'expression
BIC.expr.transposed <- t(BIC.expr)
```

Charger les étiquettes de classes des échantillons

Le fichier `BIC_sample-classes.tsv.gz`contient les étiquettes de classes des échantillons.

```
BIC.sample.classes <- read.table(
  file.path(data.folder, "BIC_sample-classes.tsv.gz"),
  header = TRUE)

kable(BIC.sample.classes[1:10,])
```

Chaque échantillon a été assigné à une classe selon la combinaison de 3 marqueurs immunologiques:

- Estrogen Receptor 1 (ER1)
- Progesterone Receptor 1 (PR1)
- Human epidermal growth factor receptor 2 (Her2)

Utilisez

- La fonction R `summary()` pour compter le nombre de patientes positives / négatives pour chacun de ces trois marqueurs.
- La fonction R `table()` pour calculer le nombre d'échantillons de chaque type de cancer.
- La fonction R `table()` pour calculer une table de contiingence des marqueurs ER1 et PR1

```
summary(BIC.sample.classes)

table(BIC.sample.classes$cancer.type)

table(BIC.sample.classes$ER1, BIC.sample.classes$PR1)

table(BIC.sample.classes$ER1,
      BIC.sample.classes$PR1,
      BIC.sample.classes$Her2)
```

Projection ACP des échantillons

Nous allons réaliser une ACP sans mise à l'échelle.

```
## Transformation ACP
BIC.prcomp <- prcomp(BIC.expr.transposed, center = FALSE, scale. = FALSE)
names(BIC.prcomp) # check the name of the fields of the PCA object

par(mfrow = c(1,2))
## Plot de l'écart-type sur les premières composantes
barplot(BIC.prcomp$sdev[1:10], col = "#FFEEDD",
        main = "Standard deviation per PC")

## Visualisation des données
plot(BIC.prcomp, main = "Variance per PC", col="#BBDDFF")

par(mfrow=c(1,1))
```

Définissez une couleur pour chaque classe, et assignez à chaque échantillon la couleur correspondant à sa classe. Dessinez ensuite un nuage de points avec les coordonnées de chaque échantillon dans les 1ère et 2ème composantes (PC2 vs PC1)

```
# Assign a color to each cancer type
classes <- unique(BIC.sample.classes$cancer.type)
class.colors <- rainbow(n = length(classes))
names(class.colors) <- classes
```

```
data.frame(class.colors)

# Associate a color to each sample accordinig to its cancer type
sample.colors <- class.colors[BIC.sample.classes$cancer.type]
table(BIC.sample.classes$cancer.type, sample.colors)
```

Question: comment interprétez-vous les barplots des écarts-types et variances pour les premières comosantes ? A discuter pendant le cours.

Clustering hiérarchique

Calcul de la matrice de distance

Nous allons maintenant calculer la distance entre chaque paire d'échantillon, en utilsiant comme métrique le **coefficient de corrélation de Spearman**, plus adapté à ce type de données que la distance euclidienne utilisée sur les données iris durant le cours

1. Lisez l'aide de la fonction `cor`, et utilisez cette fonction pour calculer la matrice de corrélation entre échantillons.
2. transformation du corrélation de Spearman en une distance à l'aide de la trnsformation : $d = 1 - r^2$

```
BIC.cor <- cor(t(BIC.expr.transposed),
               method = "spearman")
dim(BIC.cor)
BIC.dist <- as.dist(1 - BIC.cor)
```

hclust

Faites un premier clustering hiérarchique, avec le critère d'aggrégation par défaut (lisez l'aide de la fonction `hclust()` pour savoir quelle est ce critère par défaut).

```
BIC.hclust.complete <- hclust(BIC.dist)
plot(BIC.hclust.complete, labels = F, hang = -1, las = 1,
     main = "Pearson dissimilarity, complete linkage")
```

2. faire un deuxième clustering hiérarchique, avec le critère d'aggrégation de Ward

```
BIC.hclust.ward <- hclust(BIC.dist, method = "ward.D2")
plot(BIC.hclust.ward, labels = F, hang = -1, las = 1,
     main = "Pearson dissimilarity, Ward linkage")
```

3. Redessiner les arbres de ces deux résultats de clustering en colorant les échantillons selon la classe de cancer.

```
plot(BIC.hclust.complete, labels = F, hang = -1, las = 1,
     main = "Pearson dissimilarity, complete linkage",
     col = sample.colors)
```

```
ClassDiscovery::plotColoredClusters(
  BIC.hclust.ward,
  labs=BIC.sample.classes$cancer.type,
  cols=sample.colors, cex=0.2)

legend("topright",
       legend=sort(classes),
```

```
      col = class.colors,
      text.col = class.colors, cex=0.6)
```

3. Comparer les classifications obtenues avec les règles d'agglomératioin complète et Ward, respectivement, en étudiant l'impact du nombre de clusters.

Astuces:

- Cous pouvez utiliser les commandes `rect.hclust` et `cutree` pour visualiser les clusters sur le dendrogramme, puis récupérer les clusters.

kmeans

1. faire un premier kmeans, par exemple, en prenant le nombre de groupe trouvé sur le `hclust`
2. faire une boucle pour trouver le nombre optimal de cluster, en calculant l'inertie intra totale en fonction du nombre de groupe `kmeans()$totss` [faire une boucle pour i allant de 1 à 10 `for (i in 1:10) {}`]
3. refaire le kmeans avec ce nombre optimal
4. visualiser ces groupes par exemple sur une projection des données dans le plan par PCA, à l'aide de la fonction 'plot(PCA(mon.data.frame, choix="ind", col.ind=mon.kmeans$cluster)).

```
## Run k-means clustering with 20 centers
BIC.kmeans <- kmeans(BIC.expr.transposed, centers=20)

## Report the table of the clusters
table(BIC.kmeans$cluster)

T1 = Sys.time()
I.intra = numeric(length=20)
I.intra[1] = kmeans(BIC.expr.transposed, centers=2)$totss
for (i in 2:20) {
  message("Running k-means with ", i, " centers")
  kmi <- kmeans(BIC.expr.transposed, centers=i)
  I.intra[i] <- kmi$tot.withinss
}

# Plot a curve showint the intra-cluster information as a function of the number of clusters
plot((1:20)-0.5, I.intra, type="s")
T2 = Sys.time()
Tdiff = difftime(T2,T1) ## Measure elapsed time

par(mfrow=c(1,2))
BIC.kmeans10 <- kmeans(BIC.expr.transposed, centers=10)
table(BIC.kmeans10$cluster)
plot(BIC.prcomp, choix="ind", label="none", col.ind=BIC.kmeans10$cluster)
BIC.kmeans3 <- kmeans(BIC.expr.transposed, centers=3)
table(BIC.kmeans3$cluster)
plot(BIC.prcomp, choix="ind", label="none", col.ind=BIC.kmeans3$cluster)
par(mfrow=c(1,1))

BIC.kmeans2 <- kmeans(BIC.expr.transposed, centers=2)
table(BIC.kmeans2$cluster)
plot(BIC.prcomp, choix="ind", label="none", col.ind=BIC.kmeans2$cluster)
```

Comparaisons

kmeans versus hclust

Nous allons maintenant comparer les résultats de ces deux méthodes de clustering.

1. à l'aide de la fonction `table`, calculez la matrice de confusion de vos deux clustering. Commentez.
2. à l'aide de la fonction `adjustedRand(clues)` calculez le RI et le ARI de vos clustering. Commentez.

```
par(mfrow=c(1,2))
plot(BIC.hclust.ward, labels=FALSE, hang=-1, main = "Ward")

rect.hclust(BIC.hclust.ward, k=3)

BIC.cutree3 <- cutree(BIC.hclust.ward, k=3)

plot(BIC.hclust.ward, labels = F, hang = -1, main = "complete")

rect.hclust(BIC.hclust.ward, k = 10)

BIC.cutree10 <- cutree(BIC.hclust.ward, k = 10)

table(BIC.cutree3, BIC.kmeans$cluster)
table(BIC.cutree10, BIC.kmeans$cluster)
par(mfrow = c(1,1))

clues::adjustedRand(BIC.cutree3, BIC.kmeans3$cluster)
clues::adjustedRand(BIC.cutree10, BIC.kmeans10$cluster)
clues::adjustedRand(BIC.cutree3, BIC.kmeans10$cluster)
clues::adjustedRand(BIC.cutree10, BIC.kmeans3$cluster)
```

clustering versus statut

Nous connaissons les types de cancer des différentes tumeurs, définie en combinant trois marqueurs immunologiques :

- HER2,
- ER1 (récepteur d'œstrogène)
- PR1 (récepteur de progestérone)

et nous obtenons les classes suivantes :

- Basal.like
- HER2pos
- Luminal.A
- Luminal.B

qqs tumeurs sont non classées

Vous pouvez lire les données concernant le type de cancer grâce à la fonction `read.table`, la ligne de commande est : `mes.classes <- read.table("../../xxxx/BIC_sample-classes.tsv", h=T)`. A l'aide de la fonction `summary`, déterminez le nombre de tumeurs pour chaque type de cancer

1. comparez vos résultats de clustering avec la réalité
   - par des visualisations
   - le calcul de la matrice de confusion
   - le calcul des rand index et adjusted rand index

2. commentez

```
plot(BIC.hclust.ward, labels=F, hang=0)
rect.hclust(BIC.hclust.ward, k=2)
BIC.cutree2 <- cutree(BIC.hclust.ward, k=2)
table(BIC.cutree2, BIC.sample.classes[,4])
table(BIC.kmeans2$cluster, BIC.sample.classes[,4])
```

```
# clustering versus cancer.type
table(BIC.cutree3, BIC.sample.classes[,1])
clues::adjustedRand(BIC.cutree3, as.numeric(BIC.sample.classes[,1]))
table(BIC.kmeans3$cluster, BIC.sample.classes[,1])
clues::adjustedRand(BIC.kmeans3$cluster, as.numeric(BIC.sample.classes[,1]))
table(BIC.cutree10, BIC.sample.classes[,1])
clues::adjustedRand(BIC.cutree10, as.numeric(BIC.sample.classes[,1]))
table(BIC.kmeans10$cluster, BIC.sample.classes[,1])
clues::adjustedRand(BIC.kmeans10$cluster, as.numeric(BIC.sample.classes[,1]))
```

```
# visualisation

## hclust (2 groupes) et HER2+/-
mycol=BIC.cutree2
mypch=as.numeric(BIC.sample.classes[,4])
pointsToPlot <- BIC.prcomp$ind$coord[,1:2]
plot(BIC.prcomp, choix="ind", label="none", col.ind="white")
points(pointsToPlot, col=mycol, pch=mypch)
texte.legend=c("HER2-", "HER2+", "gr1", "gr2")
legend(x=-400, y=100, texte.legend, col=c(1, 1, 1, 2), pch=c(1, 2, NA, NA), text.col=c(1, 1, 1, 2))
```