Détection de gènes différentiellement exprimés à partir de données RNA-seq

Diplôme Interuniversitaire en Bioinformatique intégrative (DU-Bii 2019)

Jacques van Helden

2019-02-24

Contents

Introduction

Depuis l'avénement du du séquençage massivement parallèle (NGS, Next Generation Sequencing) en 2007, la détection de gènes différentiellement exprimés (DEG) à partir de données transcriptomiques RNA-seq constitue l'une de ses application les plus populaires.

Le principe est de mesurer, dans différentes conditions, la concentration d'ARN correspondant à chaque gène, et de comparer ces concentrations entre les échantillons de deux (**comparaison à 2 groupes** ou **binaire**) ou plusieurs conditions (comparaison **multi-groupes**).

Dès les premières analyses, les chercheurs se sont rendus compte que les méthodes classiques d'analyse différentielles (tests de Student, ANOVA) ne se prêtaient pas du tout à l'analyse de telles données, car leur nature diffère fondamentalement des hypothèss de travail sous-jacentes, pour différentes raisons.

1. Le niveau d'expression d'un gène est quantifié sur une échelle discrète (comptage du nombre de lectures alignées qui chevauchent le gène), alors que les tests paramétriques reposent sur une hypoothèse de normalité.

2. Les ordres de grandeur des comptages varient fortement d'un gène à l'autre: certains gènes ont une poignée de comptages par échantillons, d'autres des centaines de milliers. Les gènes très très fortement représentés correspondent généralement à des gènes non-codants (par exemple ARN ribosomique) ou à des fonctions cellulaires particulières liées au métabolisme de l'ARN, et on comprend les raisons biologiques de leur sur-représentation. Ils n'en constituent pas moins ce qu'on appelle en statistique des "valeurs aberrantes" ("outliers").

3. Les distributions de comptages comptent généralement un très grand nombre de valeurs nulles ("zero-inflated distributions").

4. Ces particularités posent des problèmes particuliers pour la normalisation des librairies de comptages :

    - la présence d'outliers affecte fortement la moyenne, et de façon très instable, ce qui biaise fortement l'estimation de la tendance centrale.

- la stratégie de repli sur des estimateurs robustes, comme la médiane est contestable du fait du très grand nombre de valeurs nulles (dans certains cas, plusn d'un quart voire la moitié des gènes ont une valeur nulle pour un échantillon).

Des méthodes spécifiques ont donc été développées dès 2010 pour affronter ces difficultés particulières de la normalisation et de l'analyse différentielle des données de RNA-seq.

But de ce TP

Le but de ce TP est d'effectuer une première exploration de l'analyse différentielle des données d'expression, sur base d'un petit cas d'étude simple: l'analyse transcriptionnelle de mutants de sporulation chez la levure **Saccharomyces cerevisiae**.

Cas d'étude

... A ECRIRE: DESCRIPTION DU CAS D'ETUDE

Sources des données

Paramètres de l'analyse

Nous définissons dans une variable R (de type liste) les paramètres de l'analyse. Ceci nous permettra de reproduire ultérieurement exactement la même succession d'étapes soit en utilisant des données différentes, soit en modifiant les paramètres particuliers (design, seuils, ...).

```
## Load libraries
message("Loading libraries")
library(knitr)
library(kableExtra)
library(FactoMineR)
library(clues)
library(RColorBrewer)
library(ComplexHeatmap)

# library(corrplot)
library(FactoMineR)
library(ClassDiscovery)
# library(formattable)
```

```
parameters <- list(
  data.folder = "data/GSE89530", # dossier des données
  # data.folder = "/shared/projects/du_bii_2019/data/module3/seance5/GSE89530" # on the IFB-cluster-cor
  counts = "GSE89530_counts.tsv.gz",
  sample.descr = "GSE89530_samples.tsv.gz",
  alpha = 0.05, # seuil de significativité
  epsilon = 0.1 # pseudo-comptage pour la transformation log2
)

kable(t(data.frame(parameters)), col.names = "Parameter Value")
```

	Parameter Value
data.folder	data/GSE89530
counts	GSE89530_counts.tsv.gz
sample.descr	GSE89530_samples.tsv.gz
alpha	0.05
epsilon	0.1

Table 1: **Table de comptage de lectures (short reads) par gène.** Sélection arbitraire de quelques gènes (lignes) et échantillons (colonnes)

	Condition	Replpicate	Genotype	Time.point	Label
GSM2375291	Wild_type_4	3	Wild-type	4	Sc_WT_4h
GSM2375286	Wild_type_8	2	Wild-type	8	Sc_WT_8h
GSM2375278	Wild_type_0	1	Wild-type	0	Sc_WT_0h
GSM2375281	bdf1_Y187F_Y354F_mutant_0	1	bdf1-Y187F-Y354F mutant	0	bd1bd2mut_0

Téléchargement des données

```
## List files in the data folder
# list.files(parameters$data.folder)

## Load counts of reads per gene
counts <- read.delim(file.path(parameters$data.folder, parameters$counts), row.names = 1, header = TRUE)

## Load sample descriptions
samples <- read.delim(file.path(parameters$data.folder, parameters$sample.descr), row.names = 1, header
```

La table de comptages comporte 7478 lignes (correspondant aux gènes) et 18 colonnes (correspondant aux échantillons). Nous pouvons afficher un petit morceau de cette table, ens électionnant au hasard quelques lignes et colonnes.

```
some.genes <- sample(1:nrow(counts), size = 10, replace = FALSE)
some.samples <- sample(1:ncol(counts), size = 4, replace = FALSE)
kable(counts[some.genes, some.samples])
```

	GSM2375289	GSM2375294	GSM2375281	GSM2375278
YEL025C	631	578	461	399
YAL002W	244	231	257	259
YLR256W	2639	1808	740	865
YKL053W	5	2	2	5
RDN37-1	0	0	0	0
YDR277C	447	240	378	564
YMR306W	3664	4073	79	165
YBR238C	164	115	110	251
YGR291C	0	0	0	0
YLR370C	499	476	792	733

Le tableau GSE89530_samples.tsv.gz fournit une description de chaque échantillon.

```
# print(samples[some.samples, ])
kable(samples[some.samples, ],
      caption = "**Table de comptage de lectures (short reads) par gène.** Sélection arbitraire de quelq
      align = "c")
```

Exploration des données

Avant toute autre chose, il convient de mener une exploration préliminaire des données, afin de se familiariser avec leur distribution.

Statistiques descriptives

La fonction R `summary()` calcule des statistiques de base pour chaque colonne d'une matrice ou data frame. Nous imprimons ici un sous-ensemble de ces échantillons.

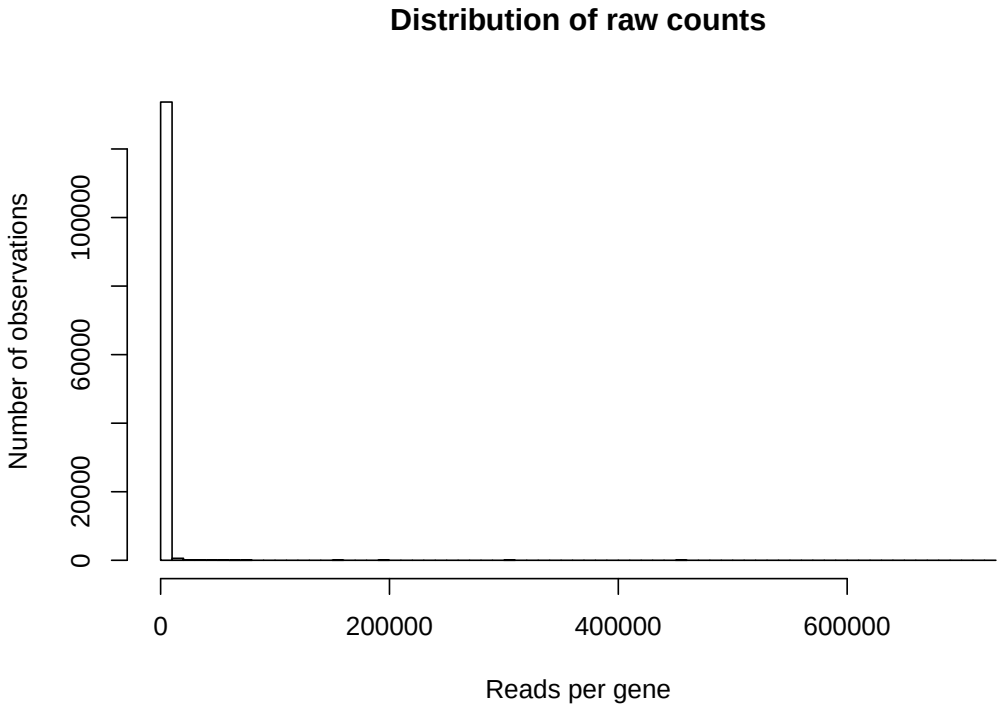
```
summary(counts[, some.samples])
```

```
   GSM2375289         GSM2375294         GSM2375281       GSM2375278
 Min.   :     0.0   Min.   :     0.0   Min.   :     0   Min.   :     0
 1st Qu.:    34.0   1st Qu.:    27.0   1st Qu.:    23   1st Qu.:    32
 Median :   248.0   Median :   208.0   Median :   253   Median :   297
 Mean   :   955.7   Mean   :   732.3   Mean   :  1007   Mean   :  1127
 3rd Qu.:   626.0   3rd Qu.:   514.8   3rd Qu.:   684   3rd Qu.:   775
 Max.   :578443.0   Max.   :454911.0   Max.   :588022   Max.   :722346
```

Distribution des comptages

```
count.stats <- mean(unlist(counts))

## Display histogram of the raw counts
hist(unlist(counts), breaks = 100, xlab = "Reads per gene", ylab = "Number of observations", main = "Dis
```



Cet histogramme n'est pas très informatif, car toutes les valeurs sont concentrées dans la première tranche (à l'extrême gauche). Ceci résulte du fiat que les intervalles de classe ont été définis sur base de l'étendue totale, et qu'il existe apparemment une observation qui a une valeur énorme par rapport aux autres (valeur aberrante, "**outlier**"). De fait, la valeur la plus élevée (722346) dépasse de très loin la moyenne (NA)

Afin de visualiser toute l'étendue des observations tout en mettant plus de détail sur les valeurs faibles, nous pouvons effectuer une transformation logarithmique. Nous devons cependant prêter attention à certains détails.

- Conventionnellement, on utlise la transformation log2 pour les données RNA-seq, car elle fournit un découpage plus fin des valeurs couvertes.

- Les données RnA-seq contiennent généralement un bon nombre de valeurs nulles (gènes non détectés), qui posent un problème pour la conversion logarithmique ($log(0) = -\infty$). On contourne ce problème en ajoutant aux comptages un petit nombre arbitraire, qu'on dénomme **pseudo-comptage** et qu'on symbolise par la lettre grècque epsilon (ϵ).

Il est cournant d'ajouter la valeur 1, mais nous préférons ajouter une valeur inférieure à 1, pour bien distinguer les comptages réels (qui peuvent de fait prendre une valeur 1) des pseudo-comptages.

Pour cette analyse nous choisissons $\epsilon = 0.1$.

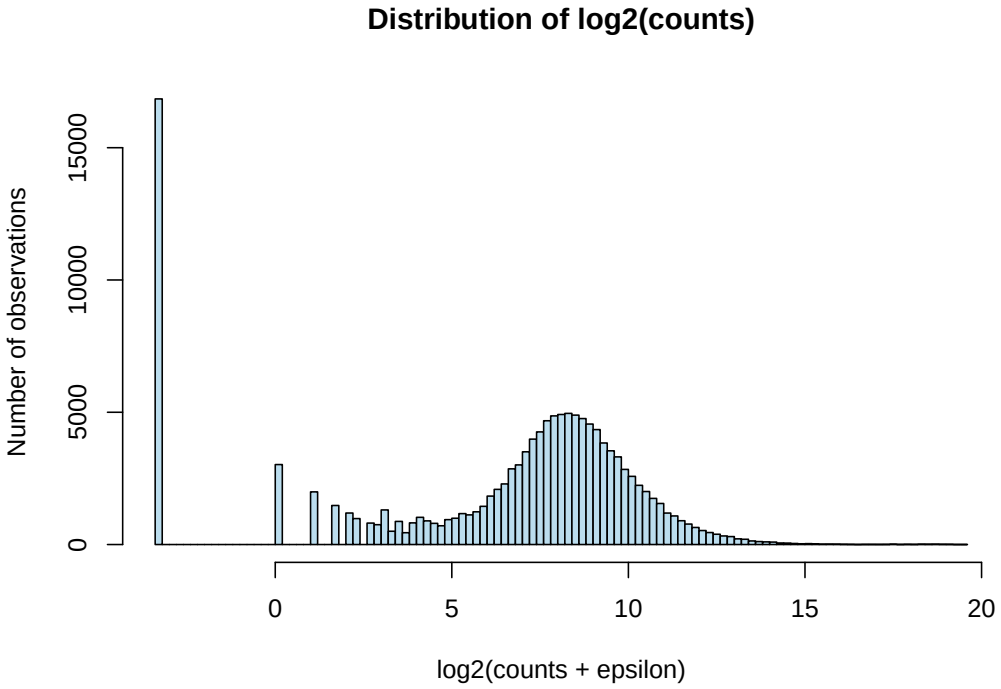
La conversion donne donc.

$$\text{log2count} = log2(n + \epsilon) = log2(n + 0.1)$$

L'histogramme des log2counts est plus informatif que celui des comptages bruts.

```
counts.log2 <- log2(counts + parameters$epsilon)

hist(unlist(counts.log2),
     xlab = "log2(counts + epsilon)",
     ylab = "Number of observations",
     main = "Distribution of log2(counts)",
     breaks = 100, col = "#BBDDEE")
```



Notons d'emblée le pic très élevé à gauche, qui correspond à toutes les observations nulles. Sur l'axe X, il apparait à la valeur -3.3, qui correspond à $log_2(\epsilon) = log_2(0.1)$.

Boîtes à moustache

Les boîtes à moustache sont très utilisées pour obtenir une vision d'ensemble de plusieurs distribution.

```
## Associate a specific color to each condition
conditions <- unique(samples$Condition)
condition.colors <- rainbow(n = length(conditions))
names(condition.colors) <- conditions
```

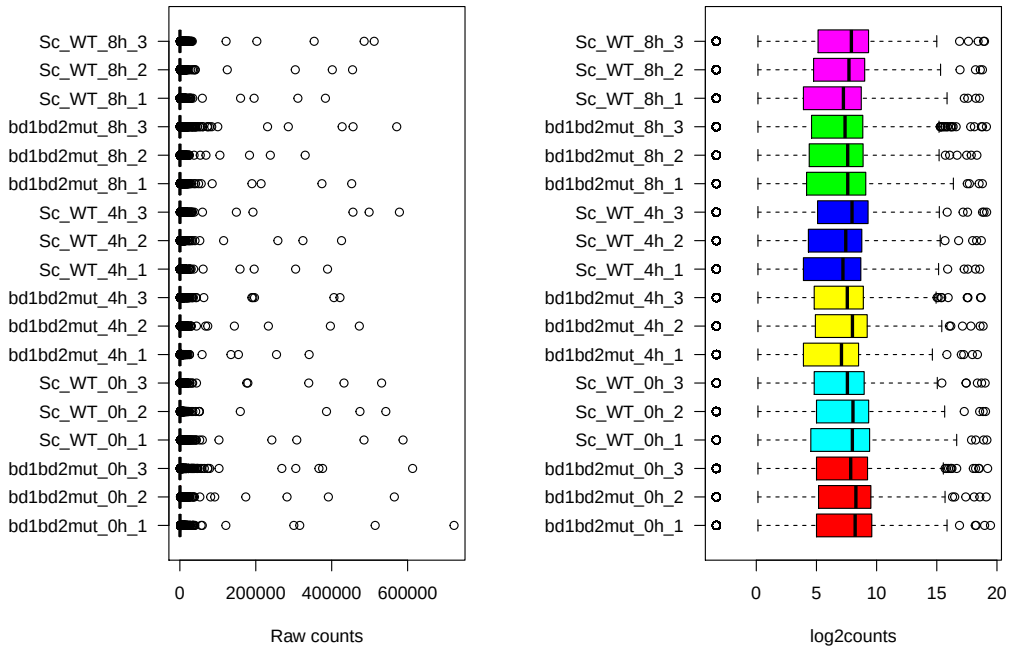


Figure 1: Boxplot of the counts per gene in each sample.

```
## Associate a color to each sample according to its condition
samples$color <- condition.colors[samples$Condition]

# Box plot of counts and log2-transformed counts
par(mfrow = c(1,2))
mar.ori <- par("mar") # Store the original value of the margin parameter
par(mar = c(5.1, 10.1, 4.1, 1.1)) # Increase leftmargin for sample labels

boxplot(counts, horizontal = TRUE, las = 1, names = samples$Label, col = samples$color, xlab = "Raw cou

boxplot(counts.log2, horizontal = TRUE, las = 1, names = samples$Label, col = samples$color, xlab = "log
```

```
par(mar = mar.ori) # reset original margins
par(mfrow = c(1,1))
```