

# TP03 - Découverte de R - Exercices

*Justine Guégan - ICONICS*

*2 juin 2017*

Ce polycopié rassemble différents exercices vous permettant d'apprendre à utiliser le langage de programmation R, en particulier, la lecture/écriture de fichiers, les graphiques, l'interprétation d'un code R existant. De manière générale, pensez à lire toutes les sous-questions car elles permettent de vous guider.

La notation entre <> n'est pas une notation R, c'est juste pour vous indiquer de personnaliser ce qu'il y a entre <>. Par exemple, `brewer.pal(4, "<NomPaletteDeVotreChoix>")` pourrait correspondre à `brewer.pal(4, "Greys")`.

Je vous propose de créer un document Rmarkdown pour ce TP.

## Exercice 1 : Résumer une liste de variants

Cet exercice a pour but de vous faire:

- lire/écrire des fichiers
- filtrer des tableaux

Le fichier `VariantsTable.xlsx` est un tableau contenant une liste de variations génomiques (snps/indels) issues d'une capture de gènes faite par séquençage à haut débit, et ce, pour 4 patients : Sample1, Sample2, Sample3 et Sample4. Cette liste de variants est annotée avec différentes base de données représentant les différentes colonnes du fichier.

**1.** Lire le fichier `VariantsTable.csv` (séparateur tabulation `\t`)

**2.** On veut travailler sur l'impact fonctionnel des variants, c'est à dire la colonne nommée : `ExonicFunc.refGene`. On souhaite créer un tableau contenant uniquement des variants d'intérêt :

**2.1** Créer un vecteur `vec1` contenant uniquement les indices des variants des catégories suivantes pour les 4 échantillons : nonsynonymous SNV, stoploss, stopgain. *Indice `which()`, /*

**2.2** Créer un vecteur `vec2` contenant uniquement les indices des variants dont le score CADD (`CADD_phred`) est supérieur à 12.37. *Indice `which()`, >*

**2.3** Créer un vecteur `vec3` contenant les indices communs entre la question 6.1 et 6.2 *Indice `intersect()`*

**2.4** Créer un nouveau tableau `tab` contenant uniquement les variants nonsynonymous SNV, stoploss, stopgain, dont le `CADD_phred` score est `> 12.37`. Vous pouvez visualiser ce tableau dans RStudio avec `View(tab)`.

**2.5** Ecrire ce nouveau tableau sur votre ordinateur grâce à la fonction `write.table`.

## Exercice 2 : Carte de chaleur (*Heatmap*)

Cet exercice a pour but de vous faire:

- lire des fichiers
- utiliser un package permettant de représenter des cartes de chaleurs, très utiles en génomique ([exemple](#))

Le fichier `counts_normalized.txt` contient les données d'expression, issues d'une expérience de RNASeq. Il y a 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). Ces données correspondent à une table de comptages avec en ligne les gènes et en colonnes les échantillons. Ci-dessous un aperçu du tableau :

|         | TNBC1      | TNBC2      | TNBC3     | TNBC4      |
|---------|------------|------------|-----------|------------|
| SGIP1   | 142.03539  | 236.39966  | 255.4641  | 143.75921  |
| AZIN2   | 73.32584   | 116.83158  | 156.2464  | 259.95858  |
| SLC45A1 | 19.08146   | 15.94601   | 36.0180   | 14.27993   |
| NECAP2  | 585.03116  | 1402.18843 | 621.9858  | 1166.31359 |
| CLIC4   | 3540.44581 | 4761.30416 | 5634.2282 | 5451.30335 |

Pour cet exercice nous aurons besoin du package `pheatmap`. Commencez par installer ce package et chargez le dans R.

1. Chargez en mémoire la table de comptage `counts_normalized.txt` en précisant que la première colonne doit correspondre aux `rownames` du tableau (pour cela regardez l'aide de la fonction `read.table`, option `row.names`). Appelez cette matrice `count`. Vous avez à présent l'habitude, vérifiez bien le tableau (`head`), les dimensions, stats de base etc ...

2. Transformez les données en  $\log_2$  (fonction `log2`) dans une nouvelle matrice appelée `countLog2`.

3. Les lignes de commandes ci-dessous permettent de réaliser un t-test entre les condition HER2 et NBS, de corriger les pvalues issues de ce test, puis de sélectionner les gènes dérégulés ( $pvalue \leq 0.05$  et  $-2 < \log_2(\text{fold-change}) < 2$ ). Exécutez-les (ceci prend un peu de temps)

```
annot = read.delim("TP3_data/exo2/annot_sample.txt")

#création d'une fonction qui calcule la moyenne par groupe
computeMean = fonction( 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)
  }
}

#exécution de cette fonction sur nos 2 groupes d'intérêt
m_nbs = computeMean(annot$condition, count, "NBS")
m_her2 = computeMean(annot$condition, count, "HER2")
```

```

M_HER2_vs_NBS = log2(m_her2 / m_nbs)

#t-test
idxHER2 = which(annot$condition == "HER2")
idxNBS = which(annot$condition == "NBS")

pvHER2 = c()
for (i in 1:nrow(countLog2)){ # boucle pour faire le test sur tous les gènes
  ttHER2 = t.test( countLog2[i,idxHER2], countLog2[i,idxNBS])
  pvHER2 = c(pvHER2, ttHER2$p.value)
}

# correction des pvalues avec la méthode de Benjamini & Hochberg
p.adjHER2 = p.adjust(pvHER2, method = "BH")

# sélection des gènes dérégulés
idx = intersect( which(p.adjHER2 <= 0.05), which(abs(M_HER2_vs_NBS) >= 1) )
countRed = countLog2[idx,c(idxHER2, idxNBS)]

```

4. Représentez à l'aide de la fonction `pheatmap` une carte de chaleur des mesures d'expression de la matrice `countLog2`, **uniquement** pour les gènes dérégulés entre les conditions HER2 et NBS. En d'autres termes, utilisez la fonction `pheatmap` sur la matrice `countRed` du bloc de code ci-dessus. Utilisez l'option `scale='row'`.

5. On peut ajouter une ou plusieurs lignes d'annotations d'échantillons et/ou gènes très facilement avec ce package. Par exemple, on va ajouter une ligne d'annotations des groupes sur notre heatmap. Pour cela :

5.1 On commence par créer la data frame d'annotations ; il est important que les rownames de cette dataframe soient les mêmes que les noms des échantillons de votre heatmap :

5.2 Puis trouvez l'option de la fonction `pheatmap` qui permet d'ajouter l'annotation des échantillons.

6 Pour finir, une astuce bien pratique en Rmarkdown. Vous avez dû remarquer que le t-test était assez long à s'exécuter. Ceci peut être le cas de certains de vos scripts. Si vous modifiez juste une virgule dans votre document Rmarkdown, vous n'avez pas envie de repasser x minutes (ou heures !) à exécuter votre code R que vous n'avez même pas touché ! Et pourtant, de base, c'est le comportement de Rmarkdown. Heureusement, une solution existe  $\emptyset$  ! Ça s'appelle le **cache** ([documentation](#)). Pour cela, dans les options des chunks, ajoutez `cache=TRUE`.

Faites un knit puis vous verrez apparaître dans votre dossier de travail un dossier appelé `***_cache`. Refaites un knit et voyez la différence de temps d'exécution. Bien entendu, si vous modifiez une ligne dans le chunk où il y a `cache=TRUE`, Rmarkdown ré-exécute votre code.