Exercices de la session 2

Vincent Guillemot & Justine Guégan

Exercice 1 : Déclaration et indexation de vecteurs et de matrices

Déclarons tout d'abord un vecteur `a` contenant les entiers pairs de 2 à 20. Le tableau suivant présente quelques solutions qui permettent de générer un tel vecteur :

Solution 1	Solution 2	Solution 3
`seq(2, 20, by=2)`	`2*(1:10)`	`(1:20)[1:20 %% 2 ==0]`

Créons ensuite un *facteur ordonné* `b` à deux niveaux (`moins` et `plus` avec `moins` < `plus`) :

```
b <- factor( rep(c("moins", "plus"), 10), ordered=TRUE )
```

1. Utilisez la fonction `table` pour vérifier que l'on a bien 10 `moins` et 10 `plus`.
2. Représentez graphiquement le résultat de `table` avec la fonction `barplot`.

   Attention (rappel) ! L'ordre qui est utilisé lors de la création de tels facteurs est l'ordre alphabétique. Par exemple le facteur ordonné `factor(c("moins", "plus", "entre les deux"), ordered=TRUE)` n'est pas dans le bon ordre par défaut. Il faut utiliser l'argument `levels` pour contraindre l'ordre entre les différents niveaux.

3. Comparez le résultat des deux commandes suivantes : `b[a]` et `b[b=="plus"]`. Est-ce que c'est normal ? On remarque que le résultat est un facteur, mais qu'il comporte des modalités inutiles ! Utilisez la fonction `droplevels` sur l'un des deux résultats précédents pour éliminer les facteurs superflus.

Travaillons maintenant sur une matrice carrée `A` (4×4) contenant les entiers de 1 à 16 arrangés en lignes :

```
A <- matrix(1:16, 4, 4, byrow = TRUE)
```

4. Extrayez la première ligne et la première colonne de cette matrice.
5. Extrayez la matrice 2×2 qui se trouve au centre des `A` et qui contient les valeurs 6, 7, 10 et 11.
6. Faites une `image` de cette matrice en utilisant l'échelle de couleurs `rainbow`.

Exercice 2 : Papillons

Les longueurs d'ailes de 24 papillons ont été mesurées (en cm) (Zar 2010):

```
butterflies <- c( 3.3, 3.5, 3.6, 3.6, 3.7, 3.8,
                  3.8, 3.8, 3.9, 3.9, 3.9, 4.0,
                  4.0, 4.0, 4.0, 4.1, 4.1, 4.1,
                  4.2, 4.2, 4.3, 4.3, 4.4, 4.5 )
```

- Représentez côte à côte un histogramme et un diagramme de Tukey de cet échantillon de mesures.

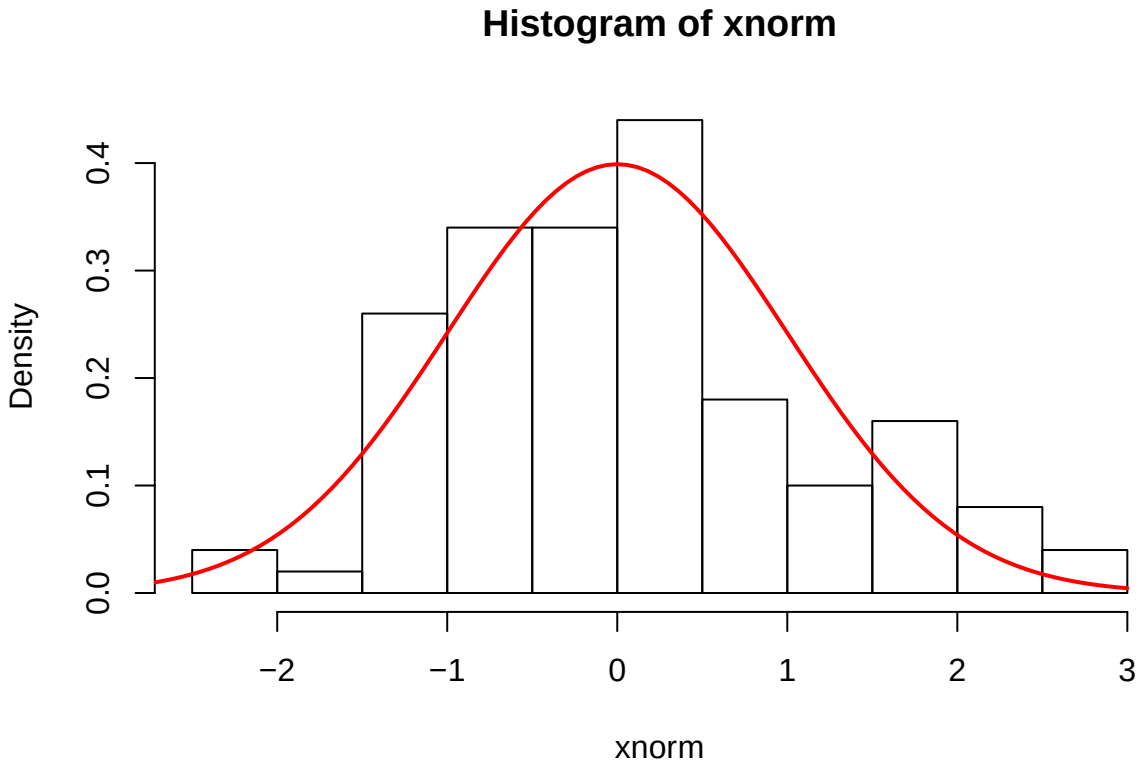
Exercice 3 : Génération de données aléatoires et représentation en histogramme

Pour chaque type de variable aléatoire suivante, tracer un histogramme d'un échantillon aléatoire de taille n.

- variable uniforme, `runif`,
- variable du Khi-deux, `rchisq`,
- variable de Fisher, `rf`,
- variable de Student, `rt`.

Un exemple pour une variable aléatoire normale vous est proposé : `hist(rnorm(100))`. Représentez également sur ce graphe la densité correspondante . Par exemple :

```
xnorm <- rnorm(100)
hist(xnorm, freq=FALSE)
plot(dnorm, col=2, lwd=2, xlim=c(-3,3), add=TRUE)
```



Exercice 4 : Création de fonction

1. Exécutez les commandes `data(iris)` puis `print(iris)`. Nous venons de charger en mémoire l'un des nombreux jeux de données distribués avec R ! Profitez de l'aide sur ce jeu de données pour en apprendre un peu plus sur les fleurs (`?iris`) !
2. Afin de représenter l'histogramme de la première colonne de `iris`, exécutez la commande `hist(iris[,1])`. Trouvez l'argument à changer de sorte que la couleur des "barres" de l'histogramme soit égale à `"steelblue"` et la couleur des bords égale à `"white"`.

3. Nous voudrions faire les histogrammes des 4 première colonnes de `iris`. Pour cela, créez une fonction `f` en suivant la syntaxe suivante

```
f <- function(i) hist(iris[,i])
```

4. Utilisez cette fonction 4 fois pour observer les histogrammes des 4 variables numériques du jeu de données `iris`.

Remarque : pour exécuter plusieurs commandes au sein d'une même fonction, il faut utiliser des accolades `{...}`. Par exemple

```
f <- function(i) {
  letitre <- paste("Histogramme de la variable", i)
  hist(iris[,i], main=letitre, xlab="Valeur")
}
```

Exercice 5 : Déclaration et indexation de tableaux et de listes

Nous nous baserons pour cet exercice sur un exemple directement inspiré des modèles linéaires. On considère le modèle linéaire suivant : on a deux mesures x et y, et un modèle qui les lie

$$y = 2x + \epsilon.$$

ϵ est un terme de bruit, que nous allons considérer pour cet exemple gaussien, de moyenne nulle et d'écart-type égal à 1.

Ce modèle ne fait pas spécialement d'hypothèses sur les distributions sous-jacentes (celles à l'origine des échantillons x et y). On peut décider, pour cet exemple, de concrétiser un peu et de se dire que x est une mesure de la "taille" d'individus (comprise uniformément entre 150 et 200 cm) et que y est une mesure du poids de ces même individus. Sans bruit, le poids d'un individu est donc calculé en multipliant sa taille par 2.

1. Générez un échantillon de taille 100 pour x à l'aide de la fonction `runif`. Attention, pour respecter la philosophie de l'exemple, il faut spécifier les bornes `min` et `max`.
2. Générez le bruit dans une variable `epsilon` avec la fonction `rnorm`.
3. Générez y en respectant le modèle spécifié ci-dessus.
4. Placez x et y dans un objet de classe `data.frame`, appelé `B`, avec la commande suivante : `B <- data.frame(x=x, y=y)`.

Pour représenter graphiquement notre expérience, nous pouvons utiliser un "biplot" :

```
plot(B, xlab="Taille (cm)", ylab="Poids (kg)")
```

Avec des *formules*, on peut reproduire le même graphique. Ce concept de formule est très utile en R. Il faut bien faire attention à la syntaxe : pour pouvoir interpréter une formule, R doit pouvoir interpréter chacun des termes. Le premier terme à gauche du signe ~ représente l'axe des ordonnées (vertical), ou encore la variable à expliquer (qui s'appelle très souvent, par convention, y). A droite du signe ~, se trouve le reste du modèle, et donc l'ensemble des variables explicatives (ici, il n'y en a qu'une, x). Pour un graphe, il ne peut y avoir qu'une variable explicative, qui représente l'axe des abscisses (horizontal).

> Comment se souvenir du sens du signe "~" ? Quand vous lisez une formule, remplacez "~" par "*en fonction de*". Ainsi, "y~x" devient "y en *fonction de* x".

Voici deux exemples de commandes qui permettent de reproduire le graphe de y *en fonction de* x :

```
plot(B$y~B$x, xlab="Taille (cm)", ylab="Poids (kg)")
plot(y~x, data=B, xlab="Taille (cm)", ylab="Poids (kg)")
```

Pour la première commande, R sait ce que vaut chaque terme de la formule `B$y~B$x` : `B$y` existe bien, il s'agit de la colonne `y` qui se trouve dans `B` ; de même pour `B$x`. En revanche, pour la deuxième expression, R ne peut pas savoir qu'il s'agit des variables `x` et `y` contenues dans `B` qu'il faut lire au moment d'interpréter la formule `y~x`. Il faut donc lui préciser grâce à l'argument `data=B`.

5. Stockez le résultat de la commande `cor.test(B$x, B$y)` dans la variable `res.test`. `res.test` est une liste ! Vérifiez le avec la fonction `mode` ! Attention, la commande `class(res.test)` donnera un résultat différent.
6. Quels sont les différents éléments qui composent cette liste ? Utilisez la fonction `names`. Récupérez dans cette liste l'intervalle de confiance (`conf.int`) de la corrélation.

Exercice 6 : Etude de la moyenne empirique sur des données simulées (Husson and Pagès 2013)

Générez, avec le logiciel R, 1000 échantillons de taille 47 et stockez les dans une matrice (`matrix`) à 1000 lignes et 47 colonnes. Ces échantillons seront gaussiens de moyenne nulle et de variance 1.

1. On veut calculer la moyenne empirique d'échantillons de taille 2, 5, 10 et 30. Utilisez les 2 premières colonnes pour calculer 1000 moyennes sur 2 individus, les colonnes 3 à 7 pour 5 individus, les colonnes 8 à 17 pour 10 individus et 18 à 47 pour 30 individus.
2. Construisez un histogramme de ces 1000 moyennes pour chaque taille d'échantillon.
3. Reprendre les mêmes questions pour une loi uniforme sur $[-1, 1]$.

Exercice 7 : Fonction de survie

19 malades atteints d'un cancer du poumon ont été traités chirurgicalement (ablation du poumon atteint) dans un même service de chirurgie et suivis jusqu'à leur décès. La série des durées de survie, mesurées en semaines a partir de la date de l'intervention chirurgicale jusqu'à celle du décès, est la suivante :

```
x <- c(25,  45, 238, 194, 16, 23, 30, 16, 22, 123, 51, 412, 162,  14, 72, 35, 30, 91, 45)
```

Dans l'étude de durée de survie, on définit la fonction de survie empirique S par $S(x) = 1 - F(x)$, où F est la fonction de répartition empirique.

1. Déterminez la médiane et les quartiles de la série statistique. Calculez l'écart interquartile.
2. Calculez la moyenne $\overline{x}$ et l'écart type $\widehat{\sigma}$ de la série.
3. Construisez le diagramme de Tukey (boite a moustaches).
4. Représentez graphiquement $F(x)$ et *attention, question difficile* $S(x)$.

Exercice 8 : Hémoglobine

On considère une série statistique de 60 taux d'hémoglobine dans le sang (gr/l) mesurés sur des adultes présumés en bonne santé.

Valeurs mesurées chez les hommes :

```
hom <- c(141 , 144 , 146 , 148 , 149 , 150 , 150 , 151 , 153 , 153 , 153 ,
         154 , 155 , 156 , 156 , 160 , 160 , 160 , 163 , 164 , 164 , 165 ,
         166 , 168 , 168 , 170 , 172 , 172 , 176 , 179.1)
```

Valeurs mesurées chez les femmes :

```
fem <- c(105 , 110 , 112 , 112 , 118 , 119 , 120 , 120 , 125 , 126 , 127 ,
         128 , 130 , 132 , 133 , 134 , 135 , 138 , 138 , 138 , 138 , 142 ,
         145 , 148 , 148 , 150 , 151 , 154 , 154 , 158)
```

1. Tracez les diagrammes de Tukey (boites a moustache) des deux échantillons.
2. Utilisez la fonction `boxplot` avec l'option `notch = TRUE` : que dit l'aide de la fonction et que pouvez vous conclure ?

Exercice 9 : Paramètres graphiques simples

1. Créez une matrice, `mat1`, composée de 100 lignes et deux colonnes; chacune des colonnes de loi normale **centrée** et de variance 1.
2. Créez une matrice, `mat2`, composée de 100 lignes et deux colonnes; chacune des colonnes de loi normale de **moyenne 1** et de variance 1.
3. Tracez le graphe bivarié de la première colonne de `mat1` sur la deuxième colonne de `mat1`.
4. Ajoutez au graphe précédent (et d'une couleur différente ; pour connaître la liste des couleurs disponibles, tapez la commande `colors()`) le graphe bivarié de la première colonne de `mat2` sur la deuxième colonne de `mat2`. Que constatez vous?
5. Rectifiez les problèmes que vous constatez à la question 4.

Exercice 10 : Faillite d'entreprises

1. Importez dans une variable nommée `G` le jeu de données nommé `faillite.csv`.
2. Quel est le mode de ce jeu de données ?
3. Convertissez `G` en une matrice `H`.
4. Tapez la commande `plot(H)`. Que construit la fonction `plot()` dans le cadre des matrices ?
5. Tapez la commande `plot(G)`. Que construit la fonction `plot()` dans le cadre d'objets de classe `data.frame` ?
6. Faites un graphique "esthétique" du jeu de données `faillite` (couleur, titre, forme, labels, etc.).

Exercice 11 : Les vins de Bordeaux

Nous allons étudier à l'aide des outils graphiques disponibles sous R le jeu de de données `bordeaux.xls`. Il s'agit d'un jeu de données qui fournit la qualité d'un vin de bordeaux (`QUALITE`) en fonction de 4 facteurs (`TEMPERAT`, `SOLEIL`, `CHALEUR`, `PLUIE`) mesurés sur 34 années consécutives.

Variable	Description
`TEMPERAT`	Somme des températures moyennes journalières (en °C)
`SOLEIL`	Durée d'insolation (en heures)
`CHALEUR`	Nombre de jours de grande chaleur
`PLUIE`	Hauteur de pluie (en mm)
`QUALITÉ`	1 = BON, 2 = MOYEN et 3 = MÉDIOCRE

On souhaite évaluer et visualiser l'impact des différents variables sur la qualité des vins de bordeaux.

1. Importer le jeu de données `bordeaux.xls`.
2. Tracer les histogrammes et les boxplots de chacune des variables.
3. Tracer les boxplots de chacune des variables conditionnellement à la qualité.

Qu'en concluez-vous?

Exercice 12 : Expression de gènes

Le jeu de données réelles sur lequel nous allons travailler dans cet exercice est extrait d'une étude sur puces à ADN qui s'intéresse à la comparaison de l'expression de gènes dans la *substance noire* chez des patients Parkinsoniens et chez des contrôles (Lesnick et al. 2007). Nous avons extrait de ce jeu de données uniquement 69 gènes qui appartiennent à trois groupes fonctionnels différents :

- `COG`, contenant des gènes liés à des processus cognitifs,
- `MOT`, contenant des gènes ayant un impact sur les fonctions motrices,
- `HLA`, un groupe contrôle n'ayant apparemment pas de lien avec la maladie de Parkinson.

1. Utilisez la fonction `load` pour charger en mémoire le jeu de données `gene.RData`.
2. Quels sont les objets qui viennent d'être chargés, et quelles en sont les caractéristiques principales ? (classe, dimensions, etc.)
3. A l'aide de la fonction `pheatmap` de la librairie du même nom, faites une représentation graphique du jeu de données.

Exercice 13 : Etude de l'expression de gènes dans certaines zones du cerveau. Visualisation et comparaison de l'expression de ces gènes

> Les données *haut-débit*, et en particulier les mesures d'expression de gènes, sont une source inépuisable d'inspiration pour les statisticiens. De nombreuses méthodes ont été ainsi dépoussiérées et adaptés au fait que l'analyse de milliers de variables pose des problèmes tout à fait différents de ce que peut être l'analyse d'une ou deux variables. Par exemple, des méthodes de visualisation comme l'*Analyse en Composantes Principales* ou les *heatmaps*, qui jusqu'à présent peinaient à prendre leur place dans la boîte à outil du statisticien, deviennent maintenant des étapes presque incoutournable de l'analyse.

Le jeu de données qui va nous servir d'exemple est extrait d'une base de données publiques et a donné lieu à la publication suivante

Altered neuronal gene expression in brain regions differentially affected by Alzheimer's disease: a reference data set. Liang, Winnie S; Dunckley, Travis; Beach, Thomas G; Grover, Andrew; Mastroeni, Diego et al. (2008) Physiological genomics vol. 33 (2) p. 240-56.

Le jeu de données lui-même est disponible sur GEO.

1. Parcourez rapidement l'article pour vous familiariser avec le jeu de données. Sur la page GEO, téléchargez le fichier `GSE5281_sample_characteristics.xls` et ouvrez le à l'aide d'un tableur.

Les données sont disponibles sont la forme d'un fichier compressé `RData` qui contient les niveaux de sondes qui sont différentiellement exprimées à la fois entre plusieurs couples de situations différentes :

```
sondes <- c("1566887_x_at", "233313_at", "241762_at", "239629_at", "237768_x_at",
            "202279_at", "202712_s_at", "205514_at", "202433_at", "201199_s_at",
            "242829_x_at", "215978_x_at", "211071_s_at", "201341_at", "242372_s_at",
            "223480_s_at", "218302_at", "201570_at", "228546_at", "238558_at",
            "219389_at", "235850_at", "226627_at", "209001_s_at", "200027_at",
            "226086_at", "201628_s_at", "230656_s_at", "200639_s_at")
```

L'obtention de ces sondes est décrite dans la table supplémentaire numéro 1 (de l'article ci-dessus).

Avant toute nouvelle analyse, il vaut mieux effacer la mémoire de travail. Cela se fait grâce à la commande `rm(list=ls())`. C'est particulièrement important lorsque l'on veut relancer une analyse *après corrections d'erreurs de programmation*. Mettez cette commande en haut de votre script.

2. Chargez en mémoire le jeu de données qui se trouve dans le fichier `expression.RData`. *Indice : `load`.* Comment s'appelle le jeu de données que nous venons de charger ? *Indice : `ls`.* Quelle est sa classe ? *Indice : `classe`.* Affichez un résumé de chacune des colonnes de ce tableau de données. *Indice : `str`.* Quelles sont ses dimensions ? *Indice : `dim`.* Quelles sont les noms des lignes et des colonnes ? *Indice : `rownames`, `colnames`.*

Un premier outil de visualisation pour des données haut-débit est le diagramme de Tukey (`boxplot`).

3. Utilisez un `boxplot` pour observer les expressions de quelques-unes (par exemple une dizaine) des puces (les colonnes) du jeu de données. Faites la même chose sur les lignes (les gènes).

Une représentation très synthétique d'un jeu de données, et qui permet d'identifier d'un coup d'oeil différentes sources de variabilités, est l'ACP (Analyse en Composantes Principales). Le principe est de résumer par quelques composantes toutes les variables d'un jeu de données. Ces composantes capturent l'une après l'autre les sources les plus importantes de variabilité.

4. Utilisez la commande suivante pour calculer l'ACP du jeu de données : `acp <- prcomp(t(expressions), scale=TRUE)`. Puis `plot(acp)` : cela permet de visualiser la variabilité capturée par chacune des composantes. Enfin, on peut représenter par des couleurs différents facteurs pour essayer de savoir s'ils corrèlent avec une ou plusieurs composantes. Exécutez la commandes `plot(acp$x, col=as.numeric(y))`. Que venons-nous de faire ? L'ACP est une source importante d'information quand on analyse des données si difficiles à manipuler : c'est un outil qui permet de repérer des sources de biais, des outliers, et de vérifier visuellement qu'un effet est présent dans un jeu de données.

Une autre représentation importante pour l'analyse de données comprenant de nombreuses variables et de nombreux individus est la *Heatmap* : elle est souvent couplée avec des méthodes de *clustering* permettant de créer de façon non-supervisée des groupes de variable et/ou d'individus. Après clustering, on voit ainsi très vite les gènes qui sont sur-exprimés ou sous-exprimés sous différentes conditions.

5. Utilisez la fonction `heatmap` sur le jeu de données `expressions` après avoir transformé celui-ci en matrice (fonction `as.matrix`). Pour changer l'échelle de couleur, utiliser la fonction `terrain.colors` de la librairie `grDevices`. Parcourez l'aide très fournie de cette fonction pour représenter le facteur `y` sur cette figure. Cela requerra l'utilisation de l'argument `RowSideColors` et de l'argument `margins`.

Références

Husson, François, and Jérôme Pagès. 2013. *Statistiques Générales Pour Utilisateurs: Exercices et Corrigés*. Presses universitaires de Rennes.

Lesnick, Timothy G, Spiridon Papapetropoulos, Deborah C Mash, Jarlath Ffrench-Mullen, Lina Shehadeh, Mariza de Andrade, John R Henley, Walter A Rocca, J Eric Ahlskog, and Demetrius M Maraganore. 2007. "A Genomic Pathway Approach to a Complex Disease: Axon Guidance and Parkinson Disease." *PLoS Genetics* 3 (6). Public Library of Science: e98.

Zar, Jerrold H. 2010. *Biostatistical Analysis*.