

TP 1 – SY02

Prise en main de R

Statistique exploratoire

Corrigé




L'objectif de ce TP est de se familiariser avec le logiciel de calcul scientifique **R** et un environnement de développement intégré **RStudio**. Nous verrons également comment calculer et manipuler les différentes statistiques vues en cours.


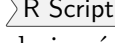
Les questions/sections marquées par un  sont des questions qui sont prévues pour être traitées en autonomie en dehors de la séance de TP.

1 Installation et prise en main

Le logiciel **R** est disponible sur une grande variété de plateformes. Il est déjà installé sur les ordinateurs des salles de TP de l'UTC. Pour l'installer sur votre ordinateur personnel, téléchargez l'archive correspondant à votre système sur le site du CRAN¹.

Nous utiliserons l'environnement de développement intégré multi-plateforme **RStudio** qui est également installé sur les ordinateurs des salles de TP de l'UTC et disponible à l'adresse <https://www.rstudio.com/products/RStudio/>

La fenêtre de démarrage du logiciel **RStudio** se compose principalement d'une console. Il s'agit d'une zone qui accepte des instructions en **R**, les exécute immédiatement en tapant **Entrée** et affiche le résultat lorsqu'il y en a un. Les instructions tapées dans cette zone *ne sont donc pas sauvegardées*. Pour éviter de retaper entièrement une commande erronée ou entrée précédemment, il existe un historique des commandes déjà entrées. Pour naviguer dans cet historique, il suffit d'utiliser les touches  et . Il est également possible de faire appel à la complétion en cours d'écriture d'une commande en appuyant sur la touche tabulation .

Pour sauvegarder vos commandes **R**, il faut créer un fichier de *script*,  **New File**  dans lequel écrire vos commandes. Pour exécuter ces instructions, on pourra au choix réexécuter la totalité du fichier en cliquant sur l'icône **Source** ou, pour plus de flexibilité, exécuter la ligne courante ou la sélection courante avec l'icône **Run** ou plus rapidement avec le raccourci clavier **ctrl**+**Entrée**.

Pour accéder à la documentation d'une fonction, par exemple **sum**, il suffit d'exécuter l'instruction **help(sum)** ou plus succinctement, **?sum**.

1. The Comprehensive R Archive Network : <https://cran.r-project.org/>

2 Premiers pas

R est un logiciel de calcul scientifique très puissant, mais il fait aussi simple calculatrice avec les opérateurs arithmétiques $+$, $*$, $-$, $/$, $**$ ainsi que les fonctions usuelles :

<pre>6*7 2**16 1+2+3+4+5 5*6/2</pre>	<pre>sqrt(36) sin(pi/15)**2 + cos(pi/15)**2 log2(2**42) log(exp(1.6))</pre>
--------------------------------------	---

① Vérifiez que :

$$\pi \approx \frac{\ln(640320^3 + 744)}{\sqrt{163}}.$$

```
(log(640320^3 + 744)/sqrt(163))
[1] 3.141593
identical(pi, log(640320^3 + 744)/sqrt(163))
[1] TRUE
```

A cause de la précision limitée de l'ordinateur, les deux expressions sont identiques. En réalité, elles diffèrent à partir de la 20^e décimale.

Pour plus de clarté dans les scripts, il est utile de stocker les résultats intermédiaires. La syntaxe en R utilise l'opérateur d'affectation `<-` pour stocker dans une variable.

```
a <- (1+sqrt(5))/2
```

La variable `a` contient désormais le nombre $\frac{1 + \sqrt{5}}{2}$ que l'on peut réutiliser :

```
a**2
a+1
```

L'opérateur `=` fonctionne également mais on préférera la notation `<-` qui est plus idiomatique.

3 Structures de données usuelles

Le langage R met à disposition des structures de données adaptées à la nature des données statistiques usuelles. Nous allons voir les trois plus simples : le vecteur, le facteur et le tableau individus-variables.

Vecteur Le vecteur est la structure de données la plus simple en R. Elle regroupe une collection de données de même type. L'instruction suivante définit ainsi un vecteur regroupant les nombres 1, 2, 3, 4, 5 dans cet ordre :

```
c(1, 2, 3, 4, 5)
```

Il s'agit en fait de la fonction `c()` (pour *concatenate*) appelée avec 5 arguments : 1, 2, 3, 4 et 5. Pour les suites de pas 1, il existe une notation encore plus simple : `1:5`.

- ② Créer le vecteur **notes** contenant les notes suivantes :

18, 1.5, 9.5, 15.5, 15, 15.5, 0.5, 14.5, 10.

```
| notes <- c(18, 1.5, 9.5, 15.5, 15, 15.5, 0.5, 14.5, 10)
```

On peut également concaténer des vecteurs. L'instruction suivante donne un résultat identique à `c(1, 2, 3, 4, 5)`.

```
| c(1, c(2, 3), c(4, 5))
```

- ③ Rajouter la note 4 au vecteur **notes**.

```
| notes <- c(notes, 4)
```

Une des forces de R est d'avoir une syntaxe simple permettant d'appliquer une même opération sur chaque élément d'un ou de plusieurs vecteurs. Tester les instructions suivantes :

<pre> v <- c(1, 2, 3, 4, 5) u <- c(5, 4, 3, 2, 1) v + 1 2*v</pre>	<pre> v/3 u * v u == 2 v > pi</pre>
--	---

- ④ On décide de ramener les notes sur 10. Créer le vecteur **notes10** des notes ramenées sur 10. Combien d'étudiants ont eu plus de 6/10 ?

```
| notes10 <- notes/2
| notes10 > 6
| [1] TRUE FALSE FALSE TRUE TRUE TRUE FALSE TRUE FALSE FALSE
```

5 étudiants ont une note supérieure à 6.

Pour extraire un sous-vecteur d'un vecteur, on utilise la notation `[]` en spécifiant les indices qui nous intéressent :

```
| v[1]
| v[2] + v[4]
| v[2:4]           # On extrait le vecteur formé par les cases 2, 3 et 4
| v[c(1, 3, 5)]    # On extrait le vecteur formé par les cases 1, 3 et 5
```

On remarquera que la numérotation des cases d'un vecteur **commence à 1** et non à 0 comme c'est le cas dans beaucoup d'autres langages.

- ⑤ Quelle est la moyenne des première, dernière et troisième notes ?

```
| (notes[1] + notes[3] + notes[10])/3
| [1] 10.5
```

On pourrait aussi utiliser

```
| mean(notes[c(1, 3, 10)])
| [1] 10.5
```

Un autre moyen d'extraction très utile est de spécifier un masque. Il s'agit d'un tableau de booléens de même taille que le vecteur qui indique si on souhaite garder un élément ou pas.

```
| choix <- c(FALSE, FALSE, TRUE, FALSE, TRUE)
| v[choix]
```

En pratique, on ne spécifie quasiment jamais le tableau de booléens mais on le crée à partir d'un vecteur de même taille en fonction des éléments que l'on veut sélectionner.

```
| choix <- v > pi
| v[choix] # On sélectionne les éléments de v supérieurs à  $\pi$ 
```

⑥ Combien de notes sont strictement supérieures à 10? On pourra utiliser la fonction `length` qui donne la longueur d'un vecteur.

On forme le masque recherché

```
| choix <- notes > 10
```

On utilise ensuite ce masque pour sélectionner les éléments de `notes`

```
| notes10 <- notes[choix]
```

On calcule enfin la longueur du vecteur extrait

```
| length(notes10)
| [1] 5
```

Ou sans faire appel à des variables intermédiaires

```
| length(notes[notes > 10])
| [1] 5
```



⑦ Quelle est la note la plus basse parmi les notes non fractionnaires? On pourra utiliser les fonctions `floor` et `min`.

Les notes non fractionnaires sont caractérisées par le fait qu'elles sont égales à leur partie entière. On forme donc le masque suivant

```
| choix <- notes == floor(notes)
```

On sélectionne les notes non fractionnaires et on calcule la valeur minimum

```
| min(notes[choix])
| [1] 4
```

Ou sans faire appel à des variables intermédiaires

```
| min(notes[notes == floor(notes)])
| [1] 4
```

Pour modifier les éléments d'un vecteur, on utilise le même opérateur d'affectation `<-` en spécifiant en plus le ou les indices comme précédemment :

```

v[2] <- 0           # On met à zéro la deuxième case du vecteur
v[2:4] <- 1         # On met à 1 les cases 2, 3 et 4
v[v == 1] <- 2      # On remplace les 1 par des 2

```

On décide de baisser toutes les notes de 2 points.

- ⑧ Créer le vecteur `notes2` des notes abaissées de 2 points.

```
notes2 <- notes - 2
```

- ⑨ Combien y a-t-il des notes négatives ? Mettez-les à zéro.

```

any(notes2 < 0)           # Y a-t-il des notes négatives?
[1] TRUE
length(notes2[notes2 < 0]) # Nombre de notes négatives
[1] 2
notes2[notes2 < 0] <- 0   # Mise à zéro des notes négatives
any(notes2 < 0)           # Y a-t-il encore des notes négatives ?
[1] FALSE

```

Facteur Les vecteurs modélisent des échantillons d'une variable **quantitative**. Pour manipuler efficacement des échantillons d'une variable **qualitative**, on utilise une variante du vecteur qu'on appelle le facteur. Pour créer un facteur à partir d'une collection de données, on utilise la fonction `factor` (les parenthèses autour d'une expression force l'affichage du résultat).

```

(collection <- c("R", "V", "B", "V"))
[1] "R" "V" "B" "V"

(f <- factor(collection))
[1] R V B V
Levels: B R V

```

On reconnaît un facteur au fait qu'il spécifie les modalités (*levels*) de la variable qualitative. La syntaxe pour l'extraction et la modification des facteurs reste la même. En revanche, comme la variable est qualitative non ordonnée, certaines opérations n'ont plus de sens.

```

2 * f
Warning in Ops.factor(2, f): '*' not meaningful for factors
[1] NA NA NA NA

f > "V"
Warning in Ops.factor(f, "V"): '>' not meaningful for factors
[1] NA NA NA NA Batman

```

Si on désire fixer un ordre entre les modalités de la variable qualitative, on peut appeler `ordered` au lieu de `factor` ou spécifier `ordered = TRUE` en argument supplémentaire de `factor`. L'ordre retenu est l'ordre alphabétique entre les modalités.

```

(f <- ordered(collection))
[1] R V B V
Levels: B < R < V

```

```
f > "B"
[1] TRUE TRUE FALSE TRUE
f <- factor(collection, ordered = TRUE)
f < "R"
[1] FALSE FALSE TRUE FALSE
```

Pour fixer un ordre différent, il faut ajouter l'argument `levels` lors de l'appel de `factor`.

```
(f <- factor(collection, ordered = TRUE, levels = c("R", "V", "B")))
[1] R V B V
Levels: R < V < B
f < "R"
[1] FALSE FALSE FALSE FALSE
```

On cherche à modéliser la séquence ADN suivante :

ACAAGATGCCATTGTC

- 10) Créer le facteur modélisant la séquence ADN. Vérifier les modalités créées ainsi que leur nombre à l'aide des fonctions `levels` et `nlevels`.

```
f <- factor(c("A", "C", "A", "A", "G", "A", "T", "G", "C", "C", "A", "T", "T", "G",
              "T", "C"))
levels(f)
[1] "A" "C" "G" "T"
nlevels(f)
[1] 4
```

- 11) Combien de nucléotides de chaque type la séquence contient-elle ?

```
length(f[f == "A"])
[1] 5
length(f[f == "T"])
[1] 4
length(f[f == "G"])
[1] 3
length(f[f == "C"])
[1] 4
```

On peut aussi utiliser `table` (voir plus loin).

Tableaux individus–variables La structure de données qui modélise un tableau individus–variables est un `data.frame`. Pour les créer, il faut spécifier les colonnes en arguments. Une colonne est représentée par un vecteur lorsque c'est une variable quantitative et par un facteur lorsque c'est une variable qualitative.

```
v <- 5:10
f <- factor(c("R", "V", "B", "R", "V", "B"))
X <- data.frame(v, f, v > 7)
X
```

```

      v f v...7
1    5 R FALSE
2    6 V FALSE
3    7 B FALSE
4    8 R  TRUE
5    9 V  TRUE
6   10 B  TRUE

```

Lors de l’affichage, R numérote les individus de la population lorsque leurs noms ne sont pas spécifiés et donne des noms aux colonnes lorsqu’ils sont absents.

En pratique, les `data.frame` peuvent être stockés dans des fichiers au format CSV². Pour les charger en mémoire, on fait appel à la fonction `read.csv` en spécifiant le chemin du fichier à charger en argument. Pour éviter de spécifier le chemin absolu du fichier, on peut changer le répertoire courant avec la commande `setwd` ou avec `Session >> Set working directory >> Choose directory` et se placer dans le répertoire contenant le fichier à charger.

12 Charger le jeu de données stocké dans le fichier `sy02.data` fourni avec le TP avec l’instruction :

```
| X <- read.csv("sy02.data")
```

Que font les fonctions `length`, `ncol`, `nrow` et `names` appliquées sur le `data.frame` `X` ?

`length` renvoie le nombre de colonnes, de même que `ncol`. `nrow` renvoie le nombre d’individus et `names` les noms des colonnes.

13 En utilisant les fonctions précédentes ainsi que les fonctions `head` ou `summary`, compter visuellement le nombre de variables qualitatives et quantitatives de ce jeu de données.

```

head(X)
  correcteur.median median correcteur.final final moyenne resultat
1              BR    11.0              ALC   17.5    14.9         C
2              EN    14.0              BR   16.0    15.2         C
3              ALC    10.5              ALC   13.0    12.0         D
4              BR    17.0              BR   13.0    14.6         C
5              EG    14.5              EN   14.0    14.2         C
6              EG    12.0              EN   19.5    16.5         B

```

Les variables `correcteur.median`, `correcteur.final`, et `resultat` sont qualitatives.
Les variables `median`, `final` et `moyenne` sont quantitatives.

```

nrow(X)
[1] 297

```

Il y a 297 individus.

Pour extraire des données d’un tableau individus-variables, on utilise une notation similaire à celle utilisée pour les vecteurs à la différence près qu’il y a désormais deux dimensions donc deux indices à spécifier, séparés par une virgule.

2. Comma-Separated Values

```

X[1,1]           # Extraire un élément
X[,3]           # Extraire la 3ème colonne
X[4,]           # Extraire la 4ème ligne
X[1:10,]        # Extraire les 10 premières lignes
X[c(1,3), c(1,4)] # Extraire les lignes 1 et 3 et les colonnes 1 et 4

```

- 14) Quelle est l'instruction permettant d'extraire la deuxième et dernière colonne du tableau individus-variables `X` ?

```

X[,2]
X[,ncol(X)]

```

Grâce à la numérotation commençant à 1, l'indice de la dernière colonne est également le nombre de colonnes.

On peut également se servir du nom des colonnes pour s'y référer. Au lieu d'écrire `X[1:10, 2]`, on peut écrire `X[1:10, 'median']`. Plus simplement, lorsqu'on veut extraire la colonne entièrement, on utilise `X$median` au lieu de `X[,2]`.

Au lieu de fournir des indices à sélectionner, on peut, comme pour les vecteurs fournir un tableau de booléens qui nous dit si l'on souhaite sélectionner l'entrée ou pas. Par exemple, l'instruction

```

X[X$median > 10,]           # Les étudiants ayant plus de 10 au médian

```

se sert du tableau de booléens `X$median > 10` pour sélectionner les étudiants ayant eu plus de 10 au médian.

- 15) Quelle est la moyenne des notes données par le correcteur `EG` lors du médian ? On pourra utiliser la fonction `mean` qui calcule la moyenne du vecteur fourni en argument.

```

mean(X[X$correcteur == "EG", "median"])
[1] 12.63208

```

- 16) Quelle est la proportion d'étudiants qui ont progressé ?

```

(nrow(X[X$median < X$final, ])/nrow(X))
[1] 0.6801347

```

4 Statistiques descriptives

Vous connaissez à présent les deux principales structures de données et comment les manipuler. Dans cette section, nous allons voir comment extraire des informations pertinentes de ces données.

- 17) Quelles sont les statistiques classiques telles que la moyenne, l'écart-type, la variance, la médiane, le maximum, le minimum et l'étendue sur les notes du final ? On utilisera les fonctions `mean`, `sd`, `var`, `median`, `max` et `min`.


```

nf <- X$final
(mean(nf))
[1] 14.75758
(sd(nf))
[1] 4.730595
(var(nf))
[1] 22.37853
(median(nf))
[1] 16
(max(nf))
[1] 20
(min(nf))
[1] 0

```

- 18) Utiliser la fonction `summary` pour retrouver quelques-unes des valeurs calculées précédemment.

```

summary(nf)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.00   12.50   16.00   14.76   18.00   20.00

```

Pour calculer les fractiles, on fait appel à la fonction `quantile`.

- 19) Quels sont les premier et troisième quartiles des notes de médian ? Calculer l'étendue inter-quartiles et retrouver-la avec la fonction `IQR`.

```

nm <- X[, "median"]
Q1 <- quantile(nm, 0.25)
Q3 <- quantile(nm, 0.75)
Q3 - Q1
75%
 6
IQR(nm)
[1] 6

```



- 20) Calculer la moyenne tronquée d'ordre 10 sur les notes du médian. On pourra utiliser la fonction `sort` qui ordonne les éléments d'un vecteur.

```

nm <- X[, "median"]
nms <- sort(nm)
l <- length(nm)
mean(nms[11:(l - 10)])
[1] 13.11372

```

5 Analyse univariée

Dans cette section, nous allons passer en revue les différentes fonctions que R met à notre disposition pour visualiser les données en fonction de leur nature.

5.1 Variables qualitatives

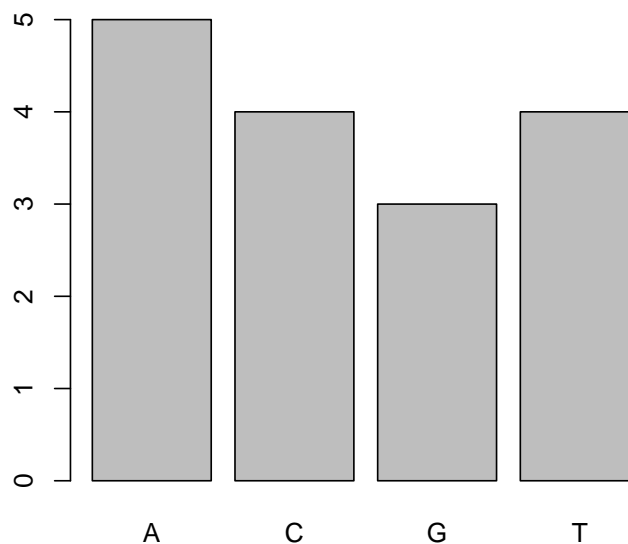
Diagramme en bandes Le diagramme en bandes permet d’afficher le nombre d’occurrences de chaque modalité dans le jeu de données considéré. Pour tracer un diagramme en bâtons en R, on va successivement faire appel à deux fonctions : `table` et `barplot`.

La première fonction, `table`, renvoie une table de contingence qui compte le nombre d’occurrences de chaque modalité. En reprenant l’exemple de la séquence ADN, on a

```
(t <- table(adn.seq))
```

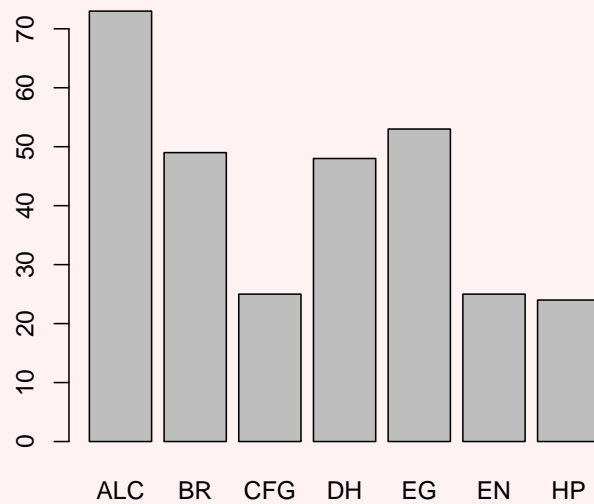
```
adn.seq  
A C G T  
5 4 3 4
```

Il suffit ensuite d’appeler `barplot` avec cette table.



- 21 Faire le diagramme en bâtons du nombre de copies de médian corrigées par correcteur. Quel est le correcteur qui a corrigé le plus de copie ?

```
counts <- table(X$correcteur.median)  
barplot(counts)
```

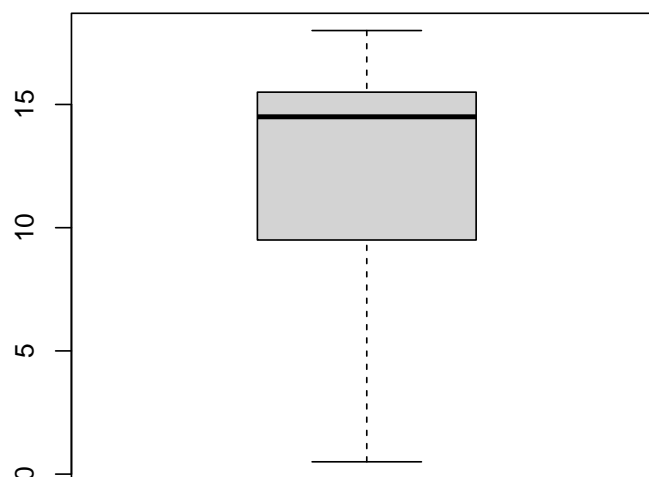


Le correcteur ALC a corrigé le plus de copie.

5.2 Variables quantitatives

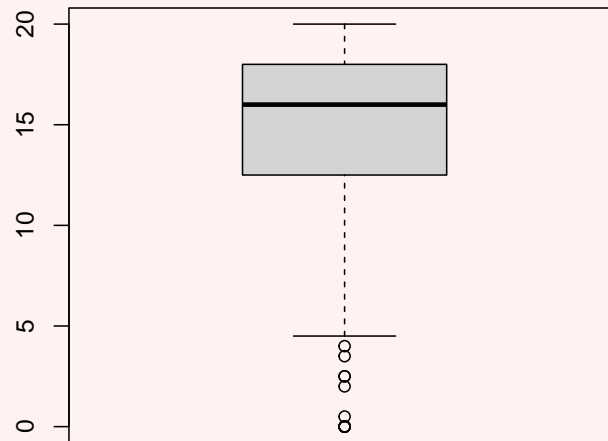
Boîte à moustaches C'est un résumé numérique d'une variable quantitative qui permet d'indiquer les statistiques renvoyées par `summary`. En R, on utilise la fonction `boxplot`. Si on reprend l'ensemble des notes de la question 2, on a

```
notes <- c(18, 1.5, 9.5, 15.5, 15, 15.5, 0.5, 14.5, 10)
boxplot(notes)
```



22 Tracer la boîte à moustaches des notes obtenues au final. Retrouver les statistiques présentes sur la boîte à moustaches avec les fonctions vues précédemment.

```
| boxplot(X$final)
```



- 23 Sachant que la moustache inférieure est la donnée immédiatement supérieure au premier quartile moins 1.5 fois l'étendue inter-quartiles, trouver le nombre de valeurs aberrantes.

```
x <- X$final
(bound <- quantile(x, 0.25) - 1.5 * IQR(x))
25%
4.25
unique(x[x < bound])
[1] 2.5 0.0 0.5 3.5 4.0 2.0
sum(x < bound)
[1] 12
```

Diagramme en tige et feuilles

- 24 Faire le diagramme en tige et feuilles de la moyenne à l'aide de la fonction `stem`.

```
| stem(X$moyenne)
```

```

The decimal point is at the |
0 | 224
1 | 247
2 | 3
3 | 22
4 | 79
5 | 0124469
6 | 01134579
7 | 02457
8 | 22355888
9 | 2224
10 | 1255
11 | 000111223345899
12 | 0000001112233335555566667777788999
13 | 0002223445566666777
14 | 00000111223333344455666666778889999
15 | 0000012223334444556666666677889999
16 | 000111223333344555666777788999
17 | 00001112233344444456667888999
18 | 0000001112222344456678889999
19 | 00011223333444678
20 | 00

```

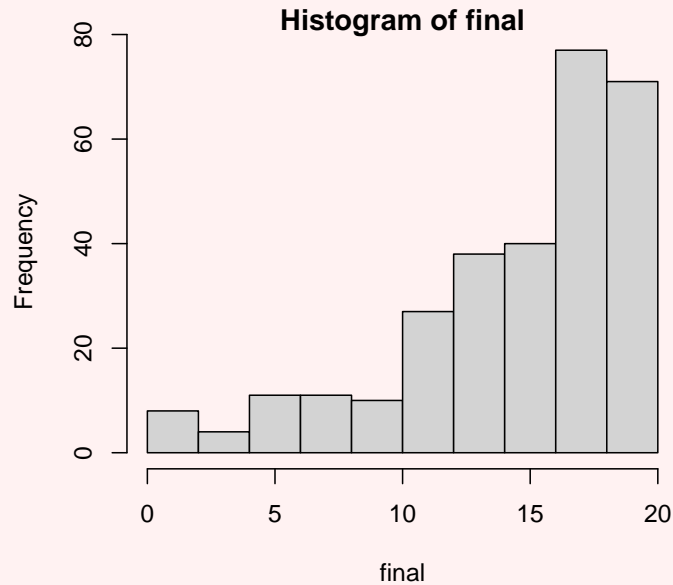
Histogramme L’histogramme permet de représenter la répartition d’un jeu de données dans un ensemble d’intervalles formant une partition. Lorsque les intervalles sont tous de même longueur, R choisit de représenter en ordonnée les effectifs. Dans le cas contraire, R calcule les ordonnées de manière à ce que l’*aire* de chaque bande de l’histogramme soit proportionnelle à l’effectif. Le coefficient de proportionnalité est fixé de manière à ce que l’aire totale soit égale à 1. Dans ce cas, l’aire d’un rectangle est exactement la proportion d’individus appartenant à l’intervalle considéré.

- 25) Utiliser la fonction `hist` pour tracer l’histogramme des notes du final.

```

final <- X$final
hist(final)

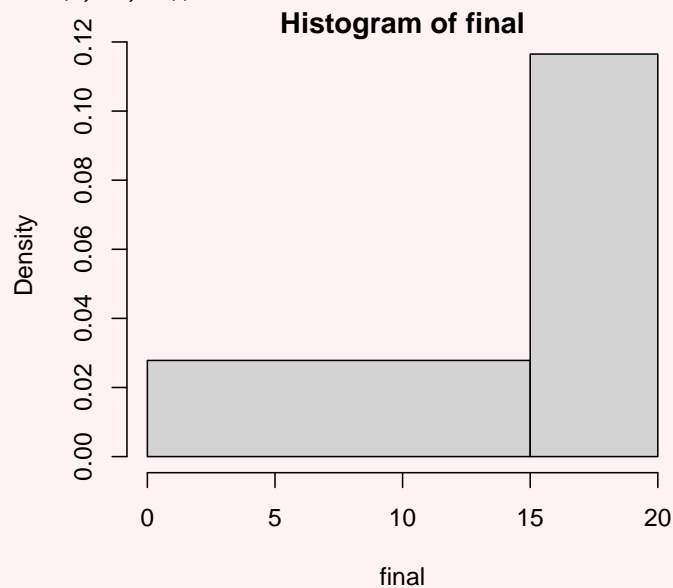
```



Noter que l'ordonnée représente ici la fréquence (`frequency`).

- 26) Dans l'histogramme, on voudrait séparer les étudiants ayant eu plus de 15 des étudiants ayant eu moins. Utiliser le paramètre `breaks` de la fonction `hist` pour spécifier un tel découpage.

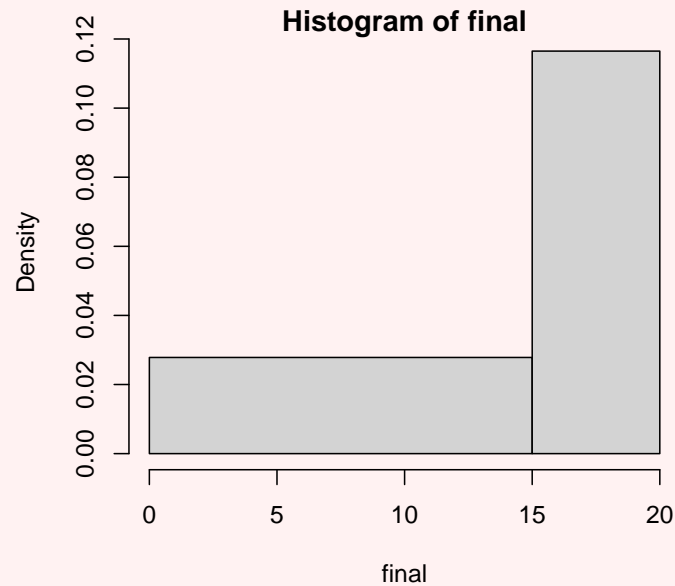
```
hist(final, breaks = c(0, 15, 20))
```



Cette fois, la subdivision n'étant pas uniforme, la densité est représentée en ordonnée.

- 27) Retrouver en R, les ordonnées des deux bandes de l'histogramme obtenu.

```
| h <- hist(final, breaks = c(0, 15, 20))
```



```
| h$density
| [1] 0.02783389 0.11649832
| length(final[final <= 15])/length(final)/15
| [1] 0.02783389
| length(final[final > 15])/length(final)/5
| [1] 0.1164983
```



(28) En utilisant la fonction `diff` et l'objet retourné par `hist`, vérifier que l'aire totale vaut 1 quel que soit le découpage spécifié par `breaks`.

```
| breaks <- c(0, 1, 10, 18, 20)
| h <- hist(final, breaks = breaks, plot = FALSE)
| sum(diff(breaks) * h$density)
| [1] 1
```

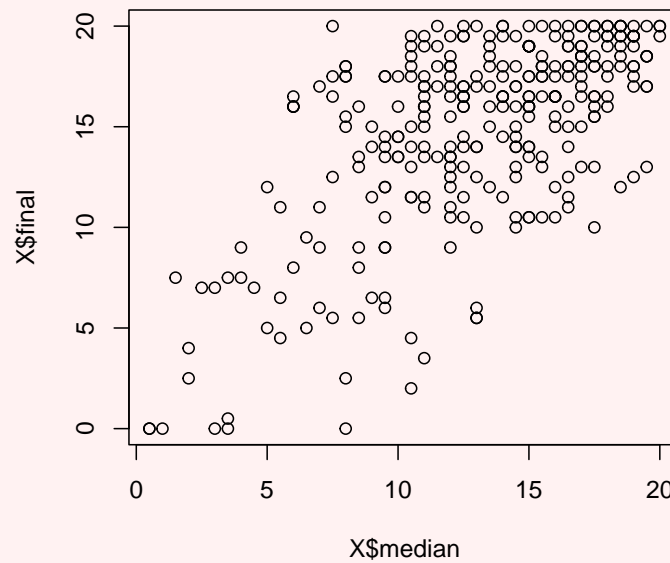
6 Analyse bivariée

6.1 Quantitative *vs* quantitative

Lorsque les variables sont quantitatives, chaque individu est caractérisé par deux nombres réels correspondant à un point dans le plan. On trace ces points en faisant appel à la fonction `plot` et en fournissant les deux vecteurs en arguments.

(29) Tracer le graphique de dispersion des notes du final par rapport à celle du médian. Que peut-on en conclure ?

```
| plot(X$median, X$final)
```



Les notes ne sont pas indépendantes. En général, les étudiants ayant réussi au médian réussissent au final.

On peut également spécifier un diagramme de manière plus souple et adaptée aux `data.frame` avec le concept de *formule*. Une formule est une écriture spéciale en R qui permet de spécifier des sorties et des entrées, séparées par le symbole `~`. Le graphique de dispersion précédent pouvait donc s'obtenir avec la commande suivante :

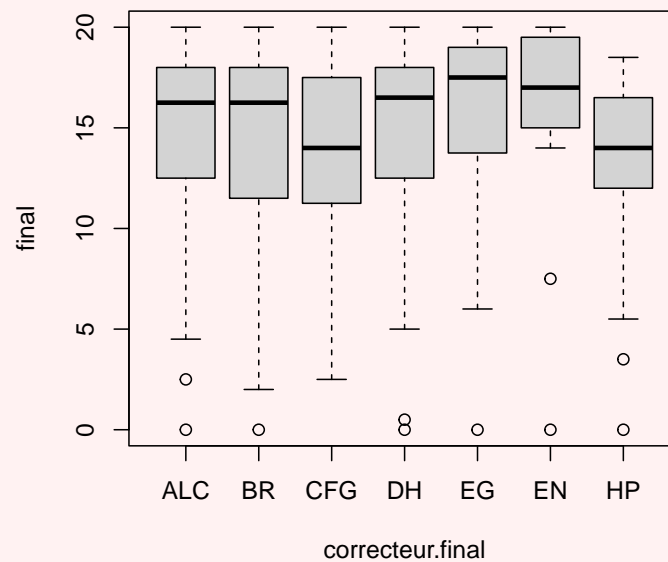
```
| plot(final ~ median, data=X)
```

On précise bien qu'on veut représenter la variable `final` (sortie) en fonction de la variable `median` (entrée) et que les informations sont à chercher dans le `data.frame` `X`.

6.2 Quantitative *vs* qualitative

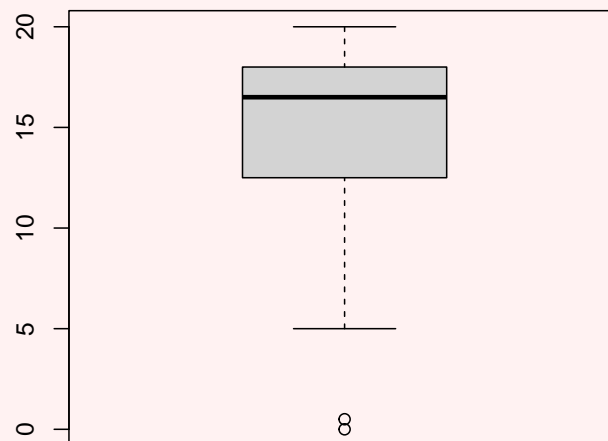
③⑩ En utilisant une formule, tracer les boîtes à moustaches des notes du final en fonction des correcteurs.

```
| boxplot(final ~ correcteur.final, data = X)
```

31) Retrouver la boîte à moustaches du correcteur DH de manière classique en filtrant les données.

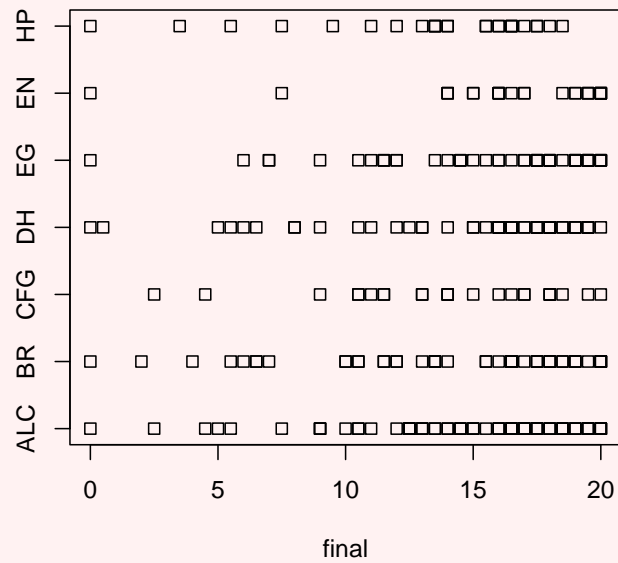
```
notes.dh <- X[X$correcteur.final == "DH", "final"]
boxplot(notes.dh)
```



Une variante lorsque la population est de taille raisonnable est de faire un graphique de dispersion unidimensionnel à la place des boîtes à moustaches. La fonction à utiliser est la fonction `stripchart`.

32) Tracer le graphique de dispersion `stripchart` des notes du final en fonction des correcteurs.

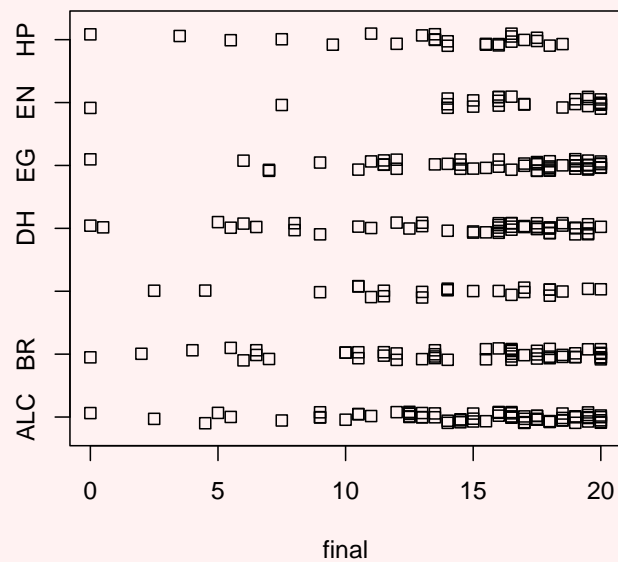
```
| stripchart(final ~ correcteur.final, data = X)
```



33) Que fait l'argument `method = "jitter"` lorsqu'il est spécifié ? Quel est l'intérêt ?

Avec `method = "jitter"`, on évite les recouvrements et on évalue mieux la densité.

```
| stripchart(final ~ correcteur.final, data = X, method = "jitter")
```



Ressources

- [1] *An Introduction to R*. URL : <https://cran.r-project.org/doc/manuals/r-release/R-intro.html>.
- [2] Peter DALGAARD. *Introductory statistics with R*. Springer Science & Business Media, 2008. URL : <http://link.springer.com/content/pdf/10.1007%2F978-0-387-79054-1.pdf>.
- [3] Vincent GOULET. *Introduction à la programmation en R*. Québec : Vincent Goulet, 2014. ISBN : 978-2-9811416-3-7. URL : https://cran.r-project.org/doc/contrib/Goulet_introduction_programmation_R.pdf.
- [4] *Learn R in Y Minutes*. URL : <https://learnxinyminutes.com/docs/fr-fr/r-fr/>.
- [5] Emmanuel PARADIS. *R pour les débutants*. 2002. URL : https://cran.r-project.org/doc/contrib/Paradis-rdebuts_fr.pdf.
- [6] Anne PHILIPPE. *Notes de cours sur le logiciel R*. Anne Philippe, 2018. URL : <https://www.math.sciences.univ-nantes.fr/~philippe/download/Anne-Philippe-cours-R.pdf>.