L2 Économie

Année 2018-2019

Module 2 - Outils Quantitatifs

Logiciel R 1

Cours et travaux dirigés

Franck Piller
Julie Scholler

université de TOURS
Faculté de droit, économie & sciences sociales

Table des matières

Première partie

Notes de cours

Présentation de l'enseignement

Enseignants :

Franck Piller : franck.piller@univ-tours.fr, bureau B246 (bâtiment B)

Julie Scholler : julie.scholler@univ-tours.fr, bureau B246 (bâtiment B)

Prérequis : Cours de statistiques de L1

Objectifs :

Le but de cet enseignement est de se familiariser avec le logiciel R et d'apprendre à utiliser les outils de gestion de données ainsi que les outils statistiques de base du logiciel R.

Plan des travaux pratiques :

- Découverte du logiciel R : interface RStudio, manipulations et objets de base ;
- Études descriptives de données enregistrées dans un *data frame* ;
- Représentations graphiques ;
- Importation, exportation et manipulation de données.

Approche et supports pédagogiques :

L'enseignement est composé de 2 séances d'une heure de cours magistral et de 5 séances de travaux pratiques de 2h. Il n'y a pas de séances toutes les semaines, surveillez votre emploi du temps.

Les sujets de TP sont disponibles sur l'ENT, ainsi que des corrections partielles.

Modalités d'évaluation :

- **Session 1 :** contrôle continu ;
- **Session 2 :** examen sous forme d'un exercice sur ordinateur.

La note de contrôle continu prendra en compte l'évaluation de trois exercices à effectuer en dehors des heures de TP et d'une épreuve sur poste informatique en fin de semestre.

Présence :

La présence en TP est obligatoire.
En cas d'absence, vous devez présenter un justificatif ou une justification au chargé de TP dans les 8 jours.

Bibliographie :

Il s'agit de lectures complémentaires aux travaux pratiques. On trouve également de nombreuses ressources sur internet :

- *Statistiques avec R*, Cornillon Pierre-André et Autres (519.5 STA) ;
- *Le logiciel R*, Lafaye de Micheaux Pierre et Autres (519.5 LAF) ;
- *Comprendre et réaliser les tests statistiques à l'aide de R*, Millot (519.5 MIL) ;
- *Initiation à la statistique avec* R, Frédéric Bertrand et Myriam Maumy-Bertrand (519.5 BER).

Thème 1

Présentation générale du logiciel de R et de R Studio

1. Présentation de R

R est un langage orienté vers le traitement de données et l'analyse statistique. Il s'agit également d'un logiciel libre publié sous licence GNU GPL.

Il permet de réaliser des analyses statistiques telles que

- des statistiques descriptives : moyenne, médiane, variance, etc. ;
- des tests d'hypothèses et des intervalles de confiance ;
- des régressions linéaires ;
- de l'analyse factorielle ;
- du machine learning ;
- des graphiques.

Avantages

- multiplateforme (Linux, Mac oS X, Windows) ;
- gratuit ;
- très puissant car les fonctionnalités de base peuvent être étendues à l'aide d'extensions (plus de 10 000) :
  - possibilités de manipulation de données supérieures à un tableur,
  - bonnes capacités graphiques et nombreuses possibilités d'export,
  - les méthodes statistiques récentes sont rapidement disponibles ;
- communauté d'utilisateurs et de développeurs très active et réactive ;
- beaucoup d'aide, d'informations et de forum à ce propos sur le web.

Inconvénients

- logiciel et documentation de base en anglais (mais de plus en plus de ressources en ligne en français)
- R s'apparente davantage à un langage de programmation qu'à un logiciel proprement dit

2. Installation de R

Sur le site `http://www.r-project.org/`, effectuer la démarche suivante :

- rubrique Download, cliquer sur CRAN ;
- choisir un site miroir en France ;
- choisir la version en fonction de votre système d'exploitation (pour Linux, il y a de fortes chances que R soit directement disponible via le gestionnaire de paquets).

3. Installation de RStudio

Sur le site `http://www.r-project.org/`, effectuer la démarche suivante :

- cliquer sur Download RStudio dans le caroussel ;
- choisir la version free de RStudio Desktop ;
- choisir la version en fonction de votre système d'exploitation (pour Linux, il y a des chances que RStudio soit directement disponible via le gestionnaire de paquets).

Thème 2

Premières manipulations

1. Principes généraux du logiciel R

La fenêtre de commandes (console) du logiciel R permet d'exécuter des instructions (commandes ou expressions).

1.1. R est une calculatrice

R permet de faire les opérations de calcul élémentaire.

Exemples :

```
> 3+5 #addition

## [1] 8

> 2*4 #multiplication

## [1] 8

> #3,5-8 # problème avec la virgule
> 3.5-8  # le bon séparateur décimal

## [1] -4.5

> # est le point
```

```
> 2.3+9-5.1

## [1] 6.2

> 3*2-5*(2-4)/6.03

## [1] 7.658375

> 3^2 # puissance

## [1] 9
```

R possède en mémoire la valeur de quelques constantes mathématiques, comme la constante π qui est appelée par la commande `pi`.

```
pi

## [1] 3.141593
```

R permet de faire des calculs plus élaborés. Il utilise pour cela des fonctions.

Exemples :

```
sqrt(2) # racine carrée

## [1] 1.414214

log(2) # logarithme népérien

## [1] 0.6931472
```

```
round(pi,2) # pi arrondi à 2 chiffres

## [1] 3.14

#après la virgule
```

1.2. Création d'objets

On peut stocker en mémoire des données, des résultats, etc. Pour cela, on définit des objets R (on reviendra sur les différents types d'objets), à l'aide du symbole `<-` qui permet d'assigner une valeur à un objet.
Exemples :

```r
a <- 5
a
```

```
## [1] 5
```

```r
b<- a+1
b
```

```
## [1] 6
```

REMARQUE : pour nommer un objet, on peut utiliser un ou plusieurs caractères alphanumériques et éventuellement des points. Cependant il faut toujours commencer par une lettre.

Certains objets existent déjà dans R. Exemple :

```r
state.area
```

```
##  [1]  51609 589757 113909  53104 158693 104247   5009   2057  58560  58876
## [11]   6450  83557  56400  36291  56290  82264  40395  48523  33215  10577
## [21]   8257  58216  84068  47716  69686 147138  77227 110540   9304   7836
## [31] 121666  49576  52586  70665  41222  69919  96981  45333   1214  31055
## [41]  77047  42244 267339  84916   9609  40815  68192  24181  56154  97914
```

Pour comprendre ces données, on peut faire appel à l'aide fournie dans R via la commande `help(state.area)`.

1.3. SCRIPT

Afin de sauvegarder son travail ou de faire des rapports, il est utile de créer des scripts. Il s'agit d'un fichier texte contenant une succession de commandes R.

Dans l'onglet File de RStudio, choisir New File puis R Script. Enregistrer votre fichier à l'emplacement de votre choix.

Pour tout travail, je vous conseille de taper directement vos commandes dans le script, puis de les exécuter dans la console en utilisant la fonction Run.

Il est possible d'insérer des commentaires dans vos scripts en les faisant précéder du caractère `#`.

```r
# ceci est un commentaire.
```

2. DONNÉES DANS R

R permet de manipuler des données organisées en structures de différentes formes (vecteurs, tableaux, etc.). Toutes ces structures sont composées d'éléments de base, ces derniers pouvant être de différents types (numériques, caractères, etc.).

2.1. MODE D'UN OBJET

Les principaux modes d'un objet de R sont :

- numeric (valeur numérique) : `1`, `pi`, `3.1416` ;
- logical (booléen, valeur logique) : `TRUE`, `FALSE`, `T`, `F` ;
- character (chaîne de caractères) : `"blabla"`.

Pour connaître le mode d'un objet `x` de R, il suffit d'exécuter la commande : `mode(x)`.
Il est possible de tester l'appartenance d'un objet à un mode en particulier avec les commandes suivantes.

```r
is.logical(a)
```

```
## [1] FALSE
```

```
is.numeric(a)
```

```
## [1] TRUE
```

```
is.character(a)
```

```
## [1] FALSE
```

Il est possible de convertir un objet d'un mode à un autre.
Exemple :

```
c <- as.character(a)
c
```

```
## [1] "5"
```

```
is.character(c)
```

```
## [1] TRUE
```

Il faut être prudent dans les conversions et bien vérifier comment R convertit.

2.2. Valeur manquante

Pour différentes raisons, il se peut que certaines données ne soient pas récoltées pendant une étude. On parle alors de données ou valeurs manquantes. Elles sont notées `NA`, pour *Not Available*. Ce n'est pas un véritable mode et il possède ses propres règles de calcul (il faudra être vigilant quand on traitera des données).
Pour savoir s'il existe une donnée manquante dans un objet `x`, il faut poser la question `is.na(x)` (`sum(is.na(x))` pour avoir le nombre de données manquantes).

2.3. Attribut d'un objet

Il y a des attributs toujours présents :

- mode : `mode` ;
- longueur : `length`.

Il y a des attributs spécifiques qui varient selon le type d'objet. On les obtient à l'aide de la commande `attributes(objet)`.

3. Première structure de données : vecteurs

C'est un objet composé d'un ensemble de valeurs toutes du même mode (numérique, logique, etc.). Le nombre d'éléments constitue l'attribut longueur.

Construction

Différentes méthodes sont possibles.

- Construction par la fonction collecteur `c()` :

```
v<-c(10,4,5,8,3,2.1,15,789,63,-2)
v
```

```
##  [1]  10.0   4.0   5.0   8.0   3.0   2.1  15.0 789.0  63.0  -2.0
```

```
c(v,12)
```

```
##  [1]  10.0   4.0   5.0   8.0   3.0   2.1  15.0 789.0  63.0  -2.0  12.0
```

```
c(TRUE,TRUE,FALSE)
```

```
## [1]  TRUE  TRUE FALSE
```

```
c(1>0,1==1, T, 1<0.5)
```

```
## [1]  TRUE  TRUE  TRUE FALSE
```

- Création par l'opérateur séquence `seq()` :

```
seq(1,8,by=0.5)
```

```
##  [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0
```

```
seq(1,2,length=5)
```

```
## [1] 1.00 1.25 1.50 1.75 2.00
```

- Création par la fonction répétition `rep` :

```
rep(1,4)
```

```
## [1] 1 1 1 1
```

```
rep("A",10)
```

```
##  [1] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
```

- encore une :

```
1:5
```

```
## [1] 1 2 3 4 5
```

Calculs

Voici quelques exemples de manipulation de vecteurs.

```
v0<- 1:5
v0 > 4
```

```
## [1] FALSE FALSE FALSE FALSE  TRUE
```

```
sqrt(v0)
```

```
## [1] 1.000000 1.414214 1.732051 2.000000 2.236068
```

```
sum(v0)
```

```
## [1] 15
```

```
sum(v0>=4)
```

```
## [1] 2
```

```
cumsum(v0)
```

```
## [1]  1  3  6 10 15
```

```
v1<-c(-3,1.2,NA,5,NA)
mode(v1)
```

```
## [1] "numeric"
```

```
is.na(v1)
```

```
## [1] FALSE FALSE  TRUE FALSE  TRUE
```

```
sum(is.na(v1))
```

```
## [1] 2
```

```
sum(v1)
```

```
## [1] NA
```

```
v1^2
```

```
## [1]  9.00  1.44    NA 25.00    NA
```

```
v2<-c(10,8,11,7,12)
```

```
sort(v2)
```

```
## [1]  7  8 10 11 12
```

```
v0+v2
```

```
## [1] 11 10 14 11 17
```

```
v1+v2
```

```
## [1]  7.0  9.2   NA 12.0   NA
```

```
v0*v2
```

```
## [1] 10 16 33 28 60
```

Sélection d'une partie d'un vecteur

On peut sélectionner une partie des éléments d'un vecteur en spécifiant les indices des termes nous intéressant. Voici quelques exemples de sélection d'une partie d'un vecteur.

```
v3<-seq(0.1,1,0.1)
v3[6] # 6ème élément du vecteur
```

```
## [1] 0.6
```

```
v3[6:8] # les éléments du vecteur des positions 6 à 8
```

```
## [1] 0.6 0.7 0.8
```

```
v3[c(1,8,3,1)]
```

```
## [1] 0.1 0.8 0.3 0.1
```

```
v3[-2] #tous les éléments de v3 sauf le deuxième
```

```
## [1] 0.1 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```
v3[-c(2,3,5:8)]
```

```
## [1] 0.1 0.4 0.9 1.0
```

On peut également sélectionner une partie d'un vecteur à l'aide d'un vecteur de valeurs logiques.

```
v3[v3>0.5] # ne renvoie que les éléments de v3 strictement supérieurs à 0.5
```

```
## [1] 0.6 0.7 0.8 0.9 1.0
```

```
v3[(v3>0.5)&(v3<1)]
```

```
## [1] 0.6 0.7 0.8 0.9
```

```
# a renvoyé les éléments de v3 strictement supérieur à 0.5 ET strictement inférieur à 1
v3[(v3<0.5)|(v3==0.9)]
```

```
## [1] 0.1 0.2 0.3 0.4 0.9
```

```
# a renvoyé les éléments de v3 strictement inférieurs à 0.5 OU égaux à 0.9
```

Autres manipulations

Recherche d'indice d'un élément :

```
v4<-c(5,1,9,7,3,4,2.5,6.32,8)
which(v4==9)
```

```
## [1] 3
```

```
which(v4>6)
```

```
## [1] 3 4 8 9
```

```
which(is.na(v1))
```

```
## [1] 3 5
```

Substitution :

```
v0[1:2]<--3
v0
```

```
## [1] -3 -3  3  4  5
```

```
v3[v3<0.5]<-0
v3
```

```
##  [1] 0.0 0.0 0.0 0.0 0.5 0.6 0.7 0.8 0.9 1.0
```

```
v1[is.na(v1)]<-1000
v1
```

```
## [1]   -3.0    1.2 1000.0    5.0 1000.0
```

```
v2[v2>=10]<-v2[v2>=10]/10
v2
```

```
## [1] 1.0 8.0 1.1 7.0 1.2
```

4. Matrices

Comme les vecteurs, une matrice est un objet composé d'un ensemble de valeurs toutes du même mode (numérique, logique, etc.). Mais les éléments sont organisés en lignes et en colonnes. Elles possèdent donc l'attribut de dimension (`dim`).
La création peut se faire à l'aide de la fonction `matrix`.

Exemples de création de matrices :

```
matrix(0,2,4)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0    0    0    0
## [2,]    0    0    0    0
```

```
matrix(1:6,nrow=2)
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
matrix(1:6,ncol=2)
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

```
matrix(1:6,ncol=2,byrow=F)
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

```
cbind(1:3,2:4)
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    2    3
## [3,]    3    4
```

```
rbind(1:3,2:4)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    2    3    4
```

5. Facteurs

Les facteurs sont des vecteurs particuliers permettant le traitement des données qualitatives. Les facteurs sont de 2 types (comme les variables qualitatives) :

- non ordonnés, par exemple mâle et femelle, appelés `factor` ;
- ordonnés, par exemple grand, moyen, petit, appelés `ordered`.

Création :

- directement par la fonction `factor` :

```
factor(c(1,22,1,1,2,2))
```

```
## [1] 1  22 1  1  2  2
## Levels: 1 2 22
```

```
s<-factor(c("m","f","f","m","m",NA,"f","f"))
s
```

```
## [1] m    f    f    m    m    <NA> f    f
## Levels: f m
```

- par conversion d'un vecteur :

```
as.factor(c(1,22,1,1,2,2))
```

```
## [1] 1  22 1  1  2  2
## Levels: 1 2 22
```

Un attribut des facteurs est `level`.

Exemples de commandes :

```
levels(s)
```

```
## [1] "f" "m"
```

```
table(s)
```

```
## s
## f m
## 4 3
```

```
prop.table(table(s))
```

```
## s
##         f         m
## 0.5714286 0.4285714
```

```
s2<-factor(c(1,0,0,2,1,0,NA,1,1,2))
levels(s2)
```

```
## [1] "0" "1" "2"
```

```
nlevels(s2)
```

```
## [1] 3
```

```
table(s2)
```

```
## s2
## 0 1 2
## 3 4 2
```

6. Tableau de données

Un *data frame* est un tableau de données dont toutes les colonnes doivent avoir la même longueur, mais peuvent être de modes différents. C'est la structure de base de R pour le traitement des données statistiques. La création d'un *data frame* peut s'effectuer en combinant des vecteurs.
Création d'un *data frame* concernant 5 étudiants de L2 Économie contenant leurs choix de module 4 et leur moyenne en L1 :

```
mod<-c("Eco","Eco","MCI","Eco","MCI")
moy<-c(14.2,11.8,12.5,10.4,13)
df<-data.frame(module=mod,moyL1=moy)
```

Observons le tableau de données construit avec les commandes suivantes :

```
df
```

```
##   module moyL1
## 1    Eco  14.2
## 2    Eco  11.8
## 3    MCI  12.5
## 4    Eco  10.4
## 5    MCI  13.0
```

```
str(df)
```

```
## 'data.frame': 5 obs. of  2 variables:
##  $ module: Factor w/ 2 levels "Eco","MCI": 1 1 2 1 2
##  $ moyL1 : num  14.2 11.8 12.5 10.4 13
```

```
dim(df)
```

```
## [1] 5 2
```

```
nrow(df)
```

```
## [1] 5
```

```
ncol(df)
```

```
## [1] 2
```

```
colnames(df)
```

```
## [1] "module" "moyL1"
```

Thème 3

Importation et exportation de données

1. Répertoire de travail

Le répertoire de travail est le répertoire où R lit et écrit ses fichiers.
La commande pour connaître le répertoire de travail actuel est `getwd()`. Pour le changer, on utilise la commande

```
setwd("chemin-vers-le-nouveau-répertoire")
```

On peut également le faire visuellement de la façon suivante : aller dans l'onglet file de la partie en bas à droite de RStudio, choisir le nouveau répertoire de travail en naviguant dans les dossiers, une fois dans le dossier, cliquer sur More et sélectionner Setting As Working Directory.
En pratique les données d'une expérience statistique sont enregistrées dans un fichier texte ou csv sous la forme d'un tableau.

2. Importation de données

On peut créer un tableau de données en utilisant la fonction `data.frame`, mais le plus souvent on obtient un tableau en important des données à partir d'un fichier existant car en pratique les données d'une expérience statistique sont enregistrées dans un fichier texte ou csv sous la forme d'un tableau.

2.1. Données simples : `read.table()`

La fonction `read.table` permet de lire et d'importer sous R des données stockées dans un fichier dans un format proche de celui des *data frames*, c'est-à-dire les variables sont arrangées en colonnes et les observations en ligne.

```
Dataframeobtenue <- read.table(file="fichierdesdonnees.txt")
```

Plusieurs options permettent de paramétrer précisément l'importation de données afin de pouvoir gérer différents formats de tableau.

Noms de colonnes et des individus

Souvent la première ligne indique les noms des colonnes. Pour le prendre en compte lors de l'importation de données, on rajoute l'option `header=TRUE` (cette option est par défaut codée `FALSE`). La première ligne n'est alors pas considérée comme des valeurs de mesure, mais comme des noms de variables.
La première colonne peut parfois contenir les noms ou numéros des individus. Pour prendre en compte cela, on utilise l'option `row.names=1`.

Séparateur de champs

Plusieurs caractères peuvent être utilisés pour séparer les données. Par défaut, la fonction `read.table()` suppose que le séparateur est un espace (incluant plusieurs espaces ou des tabulations). Mais d'autres caractères peuvent être utilisés comme séparateur afin de pouvoir distinguer les valeurs entre elles.
Exemple :

```
Dataframeobtenue <- read.table(file="fichierdesdonnees.txt", sep=";")
Dataframeobtenue <- read.table(file="fichierdesdonnees.txt", sep=",")
Dataframeobtenue <- read.table(file="fichierdesdonnees.txt", sep="\t") #pour la tabulation
```

Séparateur décimal

On peut spécifier le caractère utilisé dans le document comme séparateur décimal classiquement une virgule ou un point à l'aide de l'option `dec`.

2.2. Données issues d'un tableur : fonctions read.csv() ou read.csv2()

Quand on souhaite utiliser des données provenant d'un tableur, on commence par les enregistrer au format csv, puis on les importe à l'aide de la fonction `read.csv` ou de la fonctions `read.csv2`. Ces deux fonctions sont identiques à `read.table` sauf pour les options par défaut.

- Pour `read.csv()`, le séparateur de champs est une virgule, le séparateur décimal est un point et par défaut, `header=TRUE`.
- Pour `read.csv2()`, le séparateur de champ est un point-virgule, le séparateur décimal est une virgule et par défaut, `header=TRUE`.

2.3. Importation assistée

Au lieu de taper à la main la ligne de commande permettant l'importation, on peut se faire assister par R. Pour cela, on clique sur l'onglet Tools, puis Import Dataset, on choisit From Text File... Là une fenêtre s'ouvre pour choisir le fichier. Une fois le fichier choisi, une fenêtre permet de choisir visuellement les différents critères d'importation. Cependant il peut parfois y avoir des problèmes, donc il faut aussi savoir importer à la main.

3. Exportation de données

Après avoir travailler sur des données dans R, on peut souhaiter les exporter/enregistrer dans un fichier. Pour cela, on utilise la fonction `write.table`.

```
write.table(objet, file="nomfichierdesortie", options)
```

Par défaut, `write.table` crée un document avec comme séparateur de colonnes des espaces, comme séparateur décimal le point et il conservent les noms des lignes et des colonnes. Généralement, on utilise

```
write.table(objet, file="nomfichierdesortie.csv", sep = ";", dec = ".", row.names = FALSE,
            col.names = TRUE)
```

On ne conserve pas les nom de lignes car très souvent dans les *data frames*, il s'agit simplement de numéro de ligne.

4. Mise en pratique

1. Télécharger les trois fichiers de données présents sur Celene et les mettre dans un répertoire appelé TP4 et spécifier ce répertoire comme répertoire de travail.

   ```
   setwd("TP4")
   ```

   Le chemin dépend de l'emplacement du dossier TP4.
2. Créer un *data frame* appelé `euc0` à partir du fichier `euc0` à l'aide de la commande suivante.

   ```
   euc0<-read.table("euc0.txt")
   ```

   Vérifier le *data frame* obtenu à l'aide des commandes `head` et `str`.

   ```
   head(euc0,3)
   ```

   ```
   ##      V1 V2 V3 V4
   ## 1 18.25 36  1  L
   ## 2 19.75 42  1  L
   ## 3 16.50 33  1  L
   ```

```
str(euc0)
```

```
## 'data.frame': 1429 obs. of  4 variables:
##  $ V1: num  18.2 19.8 16.5 18.2 19.5 ...
##  $ V2: int  36 42 33 39 43 34 37 41 27 30 ...
##  $ V3: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ V4: Factor w/ 5 levels "D","DH","L","LD",..: 3 3 3 3 3 3 3 3 5 5 ...
```

3. Même chose avec le fichier `euc1` qui contient en plus les noms des colonnes. Trouver le bon argument pour importer correctement les données.

```
euc1<-read.table("euc1.txt")
head(euc1,3)
```

```
##        V1   V2   V3    V4
## 1 hauteur circ bloc clone
## 2   18.25   36    1     L
## 3   19.75   42    1     L
```

Les titres de colonnes ne sont pas pris en compte. Pour que ce soit le cas, il faut utiliser l'argument scriptheader.

```
euc1<-read.table("euc1.txt",header=TRUE)
head(euc1,3)
```

```
##   hauteur circ bloc clone
## 1   18.25   36    1     L
## 2   19.75   42    1     L
## 3   16.50   33    1     L
```

```
str(euc1)
```

```
## 'data.frame': 1429 obs. of  4 variables:
##  $ hauteur: num  18.2 19.8 16.5 18.2 19.5 ...
##  $ circ   : int  36 42 33 39 43 34 37 41 27 30 ...
##  $ bloc   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ clone  : Factor w/ 5 levels "D","DH","L","LD",..: 3 3 3 3 3 3 3 3 5 5 ...
```

4. Même chose avec le fichier `euc2`.

```
euc2<-read.table("euc2.txt",header=TRUE)
head(euc2,3)
```

```
##   hauteur.circ.bloc.clone
## 1            18.25;36;1;L
## 2            19.75;42;1;L
## 3             16.5;33;1;L
```

```
str(euc2)
```

```
## 'data.frame': 1429 obs. of  1 variable:
##  $ hauteur.circ.bloc.clone: Factor w/ 936 levels "11.25;26;1;DH",..: 145 267 59 150 239 50
```

On ne constate qu'une seule variable. Cela est dû au séparateur de champs qui est ici un point-virgule. Il faut donc le spécifier avec l'argument `sep`.

```
euc2<-read.table("euc2.txt",header=TRUE,sep=";")
head(euc2,3)
```

```
##   hauteur circ bloc clone
## 1   18.25   36    1     L
## 2   19.75   42    1     L
## 3   16.50   33    1     L
```

```
str(euc2)
```

```
## 'data.frame': 1429 obs. of  4 variables:
##  $ hauteur: num  18.2 19.8 16.5 18.2 19.5 ...
##  $ circ   : int  36 42 33 39 43 34 37 41 27 30 ...
##  $ bloc   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ clone  : Factor w/ 5 levels "D","DH","L","LD",..: 3 3 3 3 3 3 3 3 5 5 ...
```

Thème 4

Exportation de graphiques

1. Création d'un fichier contenant un graphique

On peut créer différents types de document contenant une image créer avec R. Les formats les plus courants sont : pdf, jpeg et png.
La syntaxe de base pour la crétion d'un pdf est la suivante.

```
pdf(file="nom_souhaité_du_fichier_du_graphique.pdf")
#code de construction du graphique
dev.off()
```

Pour faire une image au format jpeg, respectivement png, il suffit de remplacer la commande `pdf` par la commande `jpeg`, respectivement `png`, et adapter l'extension du nom de fichier.

```
jpeg(file="nom_souhaité_du_fichier_du_graphique.jpg")
#code de construction du graphique
dev.off()
#
png(file="nom_souhaité_du_fichier_du_graphique.png")
#code de construction du graphique
dev.off()
```

2. Personnalisation du graphique créé

Ces commandes possèdent différentes options permettant de personnaliser le document créé.

2.1. Options générales

Les plus importantes sont `height` et `width` qui permettent de spécifier la hauteur et la largeur de l'image créée. Pour la commande `pdf`, l'unité par défaut est le pouce alors que pour les autres commandes, il s'agit du pixel.

La taille de l'écriture est controlée avec l'option `pointsize` qui par défaut vaut 12.

2.2. Spécificités pour la commande `pdf`

Lors de la création d'une image en pdf, on peut également spécifier la taille du document avec l'option `paper` qui avec les arguments `a4` et `a4r`, permet de créer un document pdf au format A4 en portrait ou en paysage.

La commande `pdf` permet également de spécifier la famille d'écriture avec l'option `family` par défaut l'écriture est en Helvetica, d'autres choix possibles sont : Bookman et Palatino.

2.3. Spécificités pour la commande `jpeg`

On contrôle la qualité de l'image créée avec l'option `quality` qui par défaut vaut 75.

```
jpeg(file="nom_souhaité_du_fichier_du_graphique.jpg",width=1920,height=1080,quality=90)
#code de construction du graphique
dev.off()
```

Deuxième partie

Travaux pratiques

TP 1

Présentation générale du logiciel R

Démarrage :

- lancer R avec RStudio ;
- créer un script qui contiendra vos commandes ;
- penser à structurer votre script en mettant les numéros des parties et des exercices en commentaire.

1. Principes généraux du logiciel R

Commençons par reprendre les commandes vues en cours magistral et dans le polycopié de cours.

1.1. R est une calculatrice

Effectuer les commandes suivantes (par colonnes) :

```
2*4                    2.3+9-5.1              sqrt(2)
3,5-8                  3*2-5*(2-4)/6.03       log(2)
3.5-8                  3^2                    round(pi,2)
```

Si vous ne comprenez pas bien les commandes, utilisez l'aide en tapant `help(sqrt)`, `help(log)` ou `help(round)`. Vous pouvez également accéder à l'aide par l'onglet `help` de la partie en bas à droite de RStudio.

Exercice 1.
Calculer avec R l'expression $e^0 + \dfrac{\sin(\pi/2)}{\sqrt{4}}$.

1.2. Création d'objets

On peut stocker en mémoire des données, des résultats, etc. Pour cela, on définit des objets R (on reviendra sur les différents types d'objets), à l'aide du symbole `<-` qui permet d'assigner une valeur à un objet.
Par exemple, taper les commandes suivantes.

```
a <- 5          b <- a+1          a          b
```

Exercice 2.
Affecter la valeur 27 à l'objet nommé `x`.
Affecter la valeur 9 à l'objet nommé `X`.
Visualiser les valeurs de `x` et de `X`.
Que constatez-vous ?
Affecter la valeur 5 à l'objet `x`. Que constatez-vous ?

Certains objets existent déjà dans R. Taper `state.area`.
Pour comprendre ces données, utiliser l'aide `help(state.area)`.

2. Les vecteurs

C'est un objet composé d'un ensemble de valeurs toutes du même mode (numérique, logique, etc.). Le nombre d'éléments constitue l'attribut longueur.

Construction

Différentes méthodes sont possibles.

- Construction par la fonction collecteur `c()` :

```
v<-c(10,4,5,8,3,2.1,15,789,63,-2)
c(v,12)
```

```
c(TRUE,TRUE,FALSE)
c(1>0,1==1, T, 1<0.5)
```

- Création par l'opérateur séquence `seq()` :

```
seq(1,8,by=0.5)
```

```
seq(1,2,length=5)
```

- Création par la fonction répétition `rep` :

```
rep(1,4)
```

```
rep("A",10)
```

- encore une :

```
1:5
```

Calculs

Voici quelques exemples de manipulation de vecteurs. Les effectuer (par colonnes).

```
v0<- 1:5
v0 > 4
sqrt(v0)
sum(v0)
sum(v0>=4)
cumsum(v0)
```

```
v1<-c(-3,1.2,NA,5,NA)
mode(v1)
is.na(v1)
sum(is.na(v1))
sum(v1)
v1^2
```

```
v2<-c(10,8,11,7,12)
sort(v2)
v0+v2
v1+v2
v0*v2
```

Sélection d'une partie d'un vecteur

On peut sélectionner une partie des éléments d'un vecteur en spécifiant les indices des termes nous intéressant. Voici quelques exemples de sélection d'une partie d'un vecteur. Les exécuter et les commenter dans le script.

```
v3<-seq(0.1,1,0.1)
v3[6]
```

```
v3[6:8]
v3[c(1,8,3,1)]
```

```
v3[-2]
v3[-c(2,3,5:8)]
```

On peut également sélectionner une partie d'un vecteur à l'aide d'un vecteur de valeurs logiques.

```
v3[v3>0.5]
```

```
v3[(v3>0.5)&(v3<1)]
```

```
v3[(v3<0.5)|(v3==0.9)]
```

Autres manipulations

Recherche d'indice d'un élément :

```
v4<-c(5,1,9,7,3,4,2.5,6.32,8)
which(v4==9)
```

```
which(v4>6)
which(is.na(v1))
```

Substitution :

```
v0[1:2]<--3
v0
v3[v3<0.5]<-0
v3
```

```
v1[is.na(v1)]<-1000
v1
v2[v2>=10]<-v2[v2>=10]/10
v2
```

Exercice 3.
Dans le script, noter les commandes répondant aux questions.

1. Créer le vecteur `u` composé de 5000 uns.
2. Créer le vecteur `v` suivant : $(1.3, 2, 5.2, 4.3, 2)$.
3. Créer le vecteur $A = (-10, -9, \ldots, 9, 10)$ que vous nommerez `vecA`, à l'aide de la commande `:`. Donner sa longueur directement à partir d'une commande sans calcul.
4. Créer le vecteur $B = (-1.5, -1.4, \ldots, 0.5)$ que vous nommerez `vecB`, à l'aide de la commande `seq()`.
5. Créer le vecteur $C = (c_1, \ldots, c_n)$ à partir de `vecA` tel que $c_i = 1$si $a_i < 0$ et $c_i = a_i$ si $a_i \geqslant 0$,

Exercice 4.
Dans le script, noter les commandes répondant aux questions. Pour la dernière question, répondre sous forme de commentaires.

1. Créer un vecteur à 20 éléments, nommé `alea`, composé de nombres aléatoires extraits d'une loi normale d'espérance 3 et d'écart type 1, en tapant la commande `rnorm(20,mean=3,sd=1)` (consulter l'aide `help(rnorm)`.
2. Afficher la valeur du cinquième élément.
3. Afficher les valeurs des 5 derniers éléments.
4. Afficher les valeurs des éléments 1, 4, 8, 12 et 18.
5. Afficher les valeurs inférieures à 2.
6. Créer un nouveau vecteur `aleabis` qui est une copie du vecteur `alea`, puis affecter la valeur 0 aux éléments inférieurs à 3 et la valeur 1 aux éléments supérieurs à 3.
7. Proposer une autre façon de répondre à la question précédente à l'aide de la fonction `ifelse()` (utiliser l'aide `help(ifelse)`).
8. Combien d'éléments de `alea` sont supérieurs à 3 ? Répéter la commande création du vecteur `alea`. Avez-vous obtenu le même nombre d'éléments supérieurs à 3 ? Pourquoi ?

3. Les facteurs

Création :

- directement par la fonction `factor` :

```
factor(c(1,22,1,1,2,2))
s<-factor(c("m","f","f","m","m",NA,"f","f"))
s
```

- par conversion d'un vecteur :

```
as.factor(c(1,22,1,1,2,2))
```

Un attribut des facteurs est `level`.

Exemples de commandes :

```
levels(s)
table(s)
prop.table(table(s))
```

```
s2<-factor(c(1,0,0,2,1,0,NA,1,1,2))
levels(s2)
nlevels(s2)
table(s2)
```

Exercice 5.
On collecte la couleur des yeux de 12 personnes.

1. Créer un facteur `couleurs` regroupant les 12 valeurs obtenues qui sont les suivantes : bleu, marron, vert, marron, marron, bleu, marron, marron, vert, vert, marron, vert.
2. Donner les commandes renvoyant le nombre de modalités de la variable étudiée et l'effectif total.
3. Donner le tableau des effectifs correspondant aux données.
4. Donner une commande renvoyant le tableau des fréquences, à l'aide des fonctions précédentes et d'un calcul.
5. Donner la commande arrondissant le tableau des fréquences au centième.

TP 2

Data frames et statistiques univariées quantitatives

Objectifs du TP :

- découvrir la classe `data.frame` ;
- réaliser des graphiques simples ;
- soigner la présentation de graphiques ;
- déterminer les paramètres d'une série statistique.

Exercice 6.
La création d'un *data frame* peut s'effectuer en combinant des vecteurs.

1. Créer le *data frame* concernant 4 individus contenant leur sexe, leur taille en *cm* et leur poids en *pounds* avec les commandes suivantes :

```
s<-c("M","F","F","F")
t<-c(182,165,159,171)
p<-c(164,115,140,147)
df<-data.frame(sexe=s,taille=t,poids=p)
```

2. Observer le tableau de données construit avec les commandes suivantes :

```
df
str(df)
dim(df)
nrow(df)
ncol(df)
colnames(df)
```

3. Construire un tableau de données nommé `data` concernant les individus précédents contenant leur sexe, leur taille en m, leur poids en kg et leur IMC (arrondi à un chiffre après la virgule). L'indice corporel s'obtient de la façon suivante :

$$\text{IMC} = \frac{\text{Poids en kg}}{\text{Taille en m}^2}$$

Exercice 7.
On va travailler sur l'objet `InsectSprays` déjà existant dans R. Il contient une variable quantitative discrète que nous allons étudier.

1. Avant de manipuler cet objet, observons différentes façons d'appeler l'aide et utilisons l'aide pour comprendre le contenu de l'objet `InsectSprays`.

```
help(insectspray)
??insectspray
help(InsectSprays)
?InsectSprays
```

2. Observer l'objet `InsectSprays` avec les commandes suivantes :

```
df<-InsectSprays
str(df)
head(df)
```

   Quelle est l'utilité de la commande `head` ?

3. Effectuer les commandes suivantes.

```
count
df$count
attach(InsectSprays)
count
```

4. Tableaux des effectifs, des fréquences et des fréquences cumulées de la variable quantitative discrète `count`.
   (a) Réaliser le tableau des effectifs de la variable `count` avec la commande suivante.

```
table(df$count)
```

(b) Réaliser le tableau des fréquences en consultant l'aide de la fonction `prop.table`. Pour plus de lisibilité, arrondir à 3 chiffres après la virgule.

(c) Construire le tableau des fréquences cumulées.

5. Représentations graphiques.

(a) Réaliser un premier graphique grâce à la commande suivante.

```
plot(df$count)
```

Essayer la commande suivante. Que permet l'argument `pch` ?

```
plot(df$count,pch=4)
```

(b) Représenter graphiquement les effectifs par modalité avec la commande suivante.

```
plot(table(df$count),main="Effectifs",xlab="Nombre d'insectes sur chaque parcelle",
     ylab="Nombre de parcelles")
```

(c) Représenter la courbe des fréquences cumulées. Utiliser l'argument `type` afin d'obtenir le graphique approprié. Faire attention aux abcisses.

(d) On peut ajouter des droites et du texte sur un graphique. Essayer les commandes suivantes.

```
plot(df$count,pch=4,main="Nombre d'insectes sur chaque parcelle",
     xlab="Numéro de parcelle",ylab="Effectif")
abline(h=10,col="red")
text(x=30,y=10,"seuil")
```

(e) Consulter l'aide de la commande `text` et utiliser l'argument adéquat pour que le mot seuil ne soit plus placé sur la droite sans modifier les coordonnées $(30, 10)$.

(f) On peut tracer plusieurs droites d'un coup avec la commande `abline`. Effectuer les commandes suivantes et essayer de les comprendre.

```
plot(df$count,pch=4,main="Nombre d'insectes sur chaque parcelle",
   xlab="Numéro de parcelle",ylab="Effectif")
abline(h=10,col="red")
text(x=30,y=10,pos=3,"seuil",col="darkred")
abline(v=12*1:5+0.5,col="blue")
text(x=12*0:5+5.25,y=rep(25,6),c("A","B","C","D","E","F"),col="blue")
```

Exercice 8.

Dans cet exercice, nous allons travailler avec le tableau de données `trees` concernant des cerisiers noirs. Il contient les informations suivantes : `Girth` (circonférence), `Height` (hauteur) et `Volume`. Les unités sont anglo-saxonnes, mais cela n'a aucune incidence pour la suite.

1. Commencer par affecter `trees` dans un nouvel objet nommé `data` et observer son contenu à l'aide de commandes déjà rencontrées.
2. Effectuer les commandes ci-dessous et les décrire en commentaire.

```
summary(data)
mean(data)
mean(data$Height)
quantile(data$Height,0.25)
IQR(data$Girth)
```

3. Trouver une commande permettant d'obtenir d'un coup tous les quartiles, puis une autre permettant d'obtenir tous les déciles.
4. Réaliser un histogramme avec la commande suivante.

```
hist(data$Height)
```

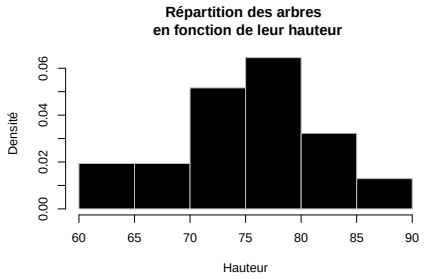
Mettre un titre principal et modifier les titres des axes.

5. Observer les différentes informations contenues dans l'objet `Histo`.

```
Histo<-hist(data$Height,plot=FALSE)
Histo
```

Que contiennent `Histo$breaks` et `Histo$counts` ?

6. Déterminer la commande qui permet d'obtenir la représentation graphique ci-contre :



7. Effectuer les commandes ci-dessous et ajouter le bon titre pour l'axe des ordonnées.

```
bornes<-c(60,70,75,80,90)
hist(data$Height,main="Répartition des arbres \n en fonction de leur hauteur",
     xlab="Hauteur",ylab="",breaks=bornes)
```

Exercice 9 (Pour évaluation).
Enregistrer chaque commande utilisée dans un script nommé VotreNom_TP2-CR.R.
Les données sont situées dans la librairie `datasets`, dans le *data frame* `Puromycin`. Ce *data frame* contient des observations concernant des vitesses de réactions enzymatiques :

- `conc` : concentration du substrat en milligramme par kilogramme ;
- `rate` : vitesse de réaction enzymatique en *counts per minute per minute* ;
- `state` : état de l'enzyme, c'est-à-dire si elle a été traitée ou non par de la puromycine, qui est un antibiotique.

1. Effectuer la commande `library(datasets)` pour charger le package et affecter les données `Puromycin` dans un *data frame* intitulé `data`.
2. Étude de la concentration de substrat.
   (a) Calculer la concentration moyenne de substrat en mg/kg. La convertir en g/kg avec cinq chiffres après la virgule.
   (b) Donner le tableau des effectifs.
   (c) Représenter graphiquement et de façon soignée et adaptée la courbe des fréquences cumulées.
3. Étude de la vitesse de réaction enzymatique.
   (a) Représenter le nuage des points de la vitesse de réaction enzymatique en fonction du numéro d'individu et ajouter sur le graphique une droite horizontale colorée correspondant au troisième quartile.
   (b) Effectuer un histogramme de la répartition de la vitesse de réaction enzymatique soigné (titres, couleurs).

TP 3

Calcul matriciel et tableaux de contingence

Objectifs du TP :

- découverte de la classe `matrix` ;
- calcul matriciel ;
- manipulation de tableaux de contingence ;
- représentations graphiques de la distribution d'une variable qualitative.

Exercice 10.
Comme un vecteur, une matrice est un objet composé d'un ensemble de valeurs toutes du même mode (numérique, logique, etc.). Mais les éléments sont organisés en lignes et en colonnes.

1. La création peut se faire à l'aide de la fonction `matrix`.
   Créer et observer les matrices `A`, `B` et `C` avec les commandes suivantes.

```
A<-matrix(1:6,nrow=2)
A
<-matrix(1:6,ncol=2)
MB
```

```
C<-matrix(1:6,ncol=2,byrow=T)
C
```

2. Tester les commandes suivantes. Que renvoient-elles ?

```
length(A)
dim(A)
```

```
length(B)
dim(B)
```

3. On peut également construire des matrices en combinant des vecteurs. Tester les commandes suivantes.

```
v1<-c(25,-12,8)
v2<-c(4.5,87,-3.2)
cbind(v1,v2)
```

4. Trouver la commande permettant de combiner `v1` et `v2` afin d'obtenir la matrice $\begin{pmatrix} 4.5 & 87 & -3.2 \\ 25 & -12 & 8 \end{pmatrix}$.

5. Réaliser les calculs suivants et noter en commentaire leurs actions.

```
A+10
A-20
A*10
```

```
A/10
A+B
B+C
```

```
A*B
B*C
A%*%B
```

```
B%*%C
A%*%t(A)
A%*%A
```

```
A^2
sqrt(MA)
```

Exercice 11 (Inverse de matrice et résolution de système).

1. Créer la matrice `A` à l'aide de la commande suivante.

```
A<-matrix(c(23,34,31,46),ncol=2)
```

2. Vérifier que la matrice `A` est inversible en calculant son déterminant. Calculer son inverse à l'aide de la commande `solve()`.
3. Vérifier que le produit $A \times A^{-1}$ est égal à la matrice identité.

4. Créer la matrice `B` égale à $\begin{pmatrix} 3 & 4 & 2 \\ 1 & 3 & 2 \\ 1 & 2 & 1 \end{pmatrix}$ et calculer le produit de `B` avec son inverse. Commenter.
5. À l'aide des commandes vues précédemment, résoudre le système suivant $\begin{cases} 23x + 31y &= 1 \\ 34x + 46y &= 2 \end{cases}$.
6. Résoudre directement l'équation de la question précédente en une seule commande à l'aide de la fonction `solve` (consulter l'aide).

Exercice 12 (Tableau de contingence).
On souhaite étudier le lien entre la couleur des cheveux et le sexe à partir d'un exemple historique dû à Fisher. Il a répertorié la couleur des cheveux de garçons et de filles d'un district écossais.

	Blond	Roux	Châtain	Brun	Noir
Garçon	592	119	849	504	36
Fille	544	97	677	451	14

1. Construire une matrice, nommée `tab`, contenant les effectifs de la même forme que le tableau précédent. Puis ajouter des noms aux lignes et aux colonnes avec les commandes suivantes.

```
rownames(tab)<-c("Garcon","Fille")
colnames(tab)<-c("Blond", "Roux","Chatain","Brun","Noir")
```

2. Sélection de parties.
   (a) Tester les deux commandes suivantes.

```
tab[2,1]
```

```
tab[1,2]
```

   (b) Extraire de `tab` un vecteur contant les effectifs des garçons roux et bruns et un autre contenant les effectifs des roux et des rousses.
3. Utiliser la commande `margin.table` pour fournir les distributions marginales en effectifs.

```
margin.table(tab,1)
```

```
margin.table(tab,2)
```

4. Construire le tableau des fréquences arrondies à 3 chiffres après la virgule.
   Ajouter les marges et changer les titres des marges de `Sum` en `Total`.
5. Donner les fréquences conditionnelles des couleurs de cheveux pour chacun des sexes.
6. Représentations graphiques.
   (a) Représenter la répartition des couleurs de cheveux des garçons avec la fonction `plot` en utilisant l'argument `lwd` pour élargir les bâtons. Utiliser également le vecteur défini ci-dessous pour colorer les différents bâtons.

```
couleur<-c("Gold","OrangeRed","Goldenrod","Brown","Black")
```

   (b) On peut également utiliser la commande `barplot` pour ce type de graphique. L'utiliser pour la répartition des couleurs de cheveux des filles, faire intervenir l'argument `space` pour espacer un peu plus les rectangles produits.
   (c) Quel est le principal avantage de la commande `barplot` ?
   (d) Une autre représentation graphique est le diagramme circulaire. Essayer la commande suivante.

```
pie(tab[1,])
```

   Travailler ce graphique en modifiant les couleurs. Essayer de trouver une façon d'ajouter les pourcentage sur le graphique.

Annexe

Exportation de graphique

Exercice 13 (Pour évaluation).
Vous allez travailler sur un jeu de données contenu dans la librairie `datasets`. Il s'agit du *data frame* `chickwts` contenant des observations sur l'efficacité de suppléments alimentaires dans l'alimentation de poulets. Ce *data frame* contient les deux variables suivantes :

- `weight` : poids des poulets en grammes après un régime de 6 semaines avec suppléments ;
- `feed` : type de suppléments : casein, horsebean, linseed, meatmeal, soybean ou sunflower.

1. Réaliser un document pdf au format A4, nommé VotreNom-graph1.pdf, contenant deux graphiques disposés l'un au-dessus de l'autre représentant chacun la répartition d'une des deux variables.
   ous devez choisir le format le mieux adapté et soigner la présentation.
2. Reproduire le graphique choisi pour la représentation de la répartition des poids des poulets mais cette fois au format jpeg. Le document produit doit s'appeler VotreNom-graph2.jpg.
   Vous devez ajouter au moins une droite et du texte, si possible de façon pertinente. Utiliser un maximum de paramètres pour soigner l'image produite.

Vous devez déposer sur Celene trois documents :

- le document VotreNom-graph1.pdf ;
- le document VotreNom-graph2.jpg ;
- le script permettant la création des ces deux documents.

TP 4

Importation de données et variables qualitatives

Objectifs du TP :

- manipulation des objets de type `factor` ;
- construction et exportation de graphiques adaptés aux données qualitatives.

Exercice 14.

1. Créer et manipuler les facteurs suivants (levels, tableaux d'effectifs).

```
f1<-factor(c(1,22,1,1,2,2))
f2<-factor(c("m","f","f","m","m",NA,"f","f"))
```

2. On collecte l'information sur la couleur des yeux de 12 personnes.
   (a) Créer un facteur `couleurs` regroupant les 12 valeurs obtenues : bleu, marron, vert, marron, marron, bleu, marron, marron, vert, vert, marron, vert.
   (b) Donner les commandes renvoyant le nombre de modalités de la variable étudiée et l'effectif total.
   (c) Construire le tableau des fréquences arrondies au centième.

Exercice 15.

Dans cet exercice, on va travailler les représentations graphiques adaptées aux situations où des variables qualitatives interviennent.
On va utiliser les données `euc2`. Les données contiennent la hauteur (`hauteur`) et la circonférence (`circ`) de différents types d'eucalyptus (`clone`) plantés dans différents sols (`bloc`).

1. Donner la structure et un résumé (moyenne, etc.) des différentes données. Commenter.
2. Attacher le *data frame*.
3. Représenter la hauteur en fonction du sol avec la commande `plot(hauteur~bloc)`. Que pensez-vous de cette représentation graphique ?
4. Les variables *type de sol* et *clone* sont-elles qualitatives ou quantitatives ? Quel est leur mode dans R ?
5. Quand on souhaite représenter une variable quantitative en fonction d'une variable qualitative, une série de diagrammes en boîte est bien adaptée. Cela se fait automatiquement avec la fonction `plot` quand la variable explicative est considérée comme un facteur. Effectuer la commande suivante.

```
plot(hauteur~clone)
```

   On peut forcer la représentation en diagramme en boîte avec la fonction `boxplot` ou faire en sorte que R considère la variable en tant que facteur.
   (a) Effectuer des diagrammes en boîte de la hauteur en fonction du type de sol à l'aide de la fonction `boxplot`.
   (b) Rajouter à la commande `boxplot` précédente les options suivantes : `range=0` et `boxwex=0.5`. Que font ces deux options ?
   (c) Changer le type de la variable `bloc` de `numeric` à `factor` avec la commande suivante.

```
bloc<-as.factor(bloc)
```

   (d) Renommer les levels de la nouvelle variable `bloc` avec la commande suivante.

```
levels(bloc)<-c("sol_A","sol_B","sol_C")
```

(e) Représenter la circonférence en fonction du type de sol avec la commande `plot`.

6. Afin de visualiser la proportion de chaque clone et de chaque type de sol, réaliser deux diagrammes circulaires.
7. Comme les clones autres que `DH` ne sont pas beaucoup représentés, nous allons fusionner tous les levels de clones autres que le level `DH` avec les commandes suivantes.

```
levels(clone)
levels(clone)<-c("autres","DH","autres","autres","autres")
str(clone)
```

8. Représenter uniquement les hauteurs pour les clones qui ne sont pas du type `DH` avec les commandes suivantes.

```
filtre_autre<-clone=="autres"
plot(hauteur[filtre_autre]~bloc[filtre_autre], xlab="Type de sol",
     ylab="Hauteur des clones de type non DH")
```

9. Enregistrer dans le *data frame* `euc2` les modifications apportées à `bloc` et `clone` avec les commandes suivantes.

```
euc2$bloc<-bloc
euc2$clone<-clone
```

10. Créer un nouveau *data frame* intitulé `eucAB` ne contenant que les individus des sols `A` et `B` et observer le nombre de levels de la variable `bloc` ainsi que son tableau d'effectifs.
11. Utiliser la commande `droplevels` pour améliorer la variable `bloc`.
12. Utiliser la commande `table` pour construire le tableau de contingence des types de sols en fonction des différents clones.

Exercice 16 (Pour évaluation).
Vous allez travailler sur les données réelles des passagers du Titanic. Elles sont contenues dans le fichier `titanic.csv` sur Celene. Ce tableau de données contient les informations suivantes :

- `survie` : codée `1` pour Oui et `0` pour Non ;
- `classe` : classe de leur cabine ;
- `sexe` : codé `F` pour les femmes et `M` pour les hommes ;
- `age` : l'âge en années ;
- `prix` : prix du ticket ;
- `embarquement` : lieu d'embarquement `C` pour Cherbourg, `Q` pour Queenstown et `S` pour Southampton.

1. Importer les données dans un *data frame* intitulé `tit`.
2. Représenter de façon adéquate la répartion des classes de cabines.
3. Représenter sur un même graphique les répartitions des prix selon le lieu d'embarquement.
4. Transformer la variable `survie` en un facteur de levels `Oui` et `Non`.
5. Représenter sur deux graphiques côte à côte la répartition des prix du ticket pour ceux ayant survécus et pour ceux n'ayant pas survécus.
6. Créer un nouveau *data frame* intitulé `titanicIB` ne contenant que les individus ayant embarqués sur les Îles Britanniques, c'est-à-dire à Southampton en Angleterre et à Queenstown en Irlande.
7. Supprimer le ou les level(s) de la variable `embarquement` devenu(s) inutile(s).
8. Exporter le tableau de données ainsi créer dans un fichier nommer `titanicIB.csv` contenant les noms des variables, ne contenant pas les noms des individus et dont le séparateur de colonnes est le point-virgule.

TP 5

Manipulations avancées de data frame

Objectifs du TP :

- extraire des individus et/ou des variables ;
- modifier/créer des variables ;
- déterminer les paramètres de sous-groupes.

Exercice 17.
Les données étudiées sont issues d'une enquête réalisée au début des années 80, concernant les étudiants américains. Elles ont été utilisées pour étudier l'effet des universités communautaires sur le niveau scolaire. Les données sont contenues dans le fichier student.txt qui contient les informations suivantes :

- `female` : 1 = *female* / 0 = *male* ;
- `black` : 1 = *black* / 0 = *not black* ;
- `hispanic` : 1 = *hispanic* / 0 = *not hispanic* ;
- `dadcoll` : 1 = père diplômé du supérieur / 0 = père non diplômé du supérieur ;
- `momcoll` : 1 = mère diplômée du supérieur / 0 = mère non diplômée du supérieur ;
- `ownhome` : 1= famille propriétaire de son logement / 0 = famille non propriétaire ;
- `urban` : 1 = école en zone urbaine / 0 = école en zone rurale ;
- `dist` : distance université-domicile en dizaine de miles ;
- `tuition` : frais de scolarité en milliers de $;
- `ed` : nombre d'années de scolarisation ;
- `incomehi` : 1 = revenu familial annuel $>$ \$25,000 / 0 = revenu familial annuel $\leqslant$ \$25,000 ;
- `county` : numéro de code du comté ;
- `state` : numéro de code de l'état.

1. Importer les données du fichier student.txt dans un *data frame* du même nom et l'observer.
2. Effectuer les commandes ci-dessous et les décrire.

```
attach(student)
mean(tuition)
mean(tuition[female==1])
mean(tuition[ed>=14])
mean(tuition[!(ed<14)])
quantile(dist[urban==1],0.25)
sd(dist[hispanic=1&tuition>1])
```

3. Déterminer la médiane du nombre d'années d'étude des femmes dont la mère est diplômée du supérieur ainsi que le troisième quartile du nombre d'années d'étude des hispaniques dont les frais de scolarité sont supérieurs à 1000 $.
4. Créer une commande qui permet de vérifier qu'aucun étudiant n'ait été codé simultanément *black* et *hispanic*.
5. Effectuer les commandes suivantes et expliquer ce qu'elles produisent.

```
tab1<-student[,c(8:10)]
head(tab1)
tab2<-student[-c(1,3),]
head(tab2)
tab3<-student[female==1,]
head(tab3)
tab4<-tab3[,-1]
head(tab4)
tab5<-student[female==1,-1]
head(tab5)
tab6<-student[county==28&tuition>=1.2,-(1:3)]
head(tab6)
```

6. Extraire dans un *data frame* intitulé `tab7` les étudiants hispaniques dont le revenu familial est supérieur à 25.000 $ et supprimer les variables `hispanic`, `black` et `incomehi`. Combien y a-t-il d'individus ?
7. À partir de cette question on va créer et transformer des variables du *data frame*. Pour éviter toute confusion entre création et transformation d'une variable, il est préférable d'annuler la commande `attach`. On effectue ceci avec la commande suivante.

   ```
   detach(student)
   ```

8. Convertir la distance université-domicile en kilomètres.
9. On souhaite créer une variable simplifiée sous forme d'un facteur codant les frais de scolarité de la façon suivante :
   - `faible` : si les frais de scolarité sont inférieurs ou égaux à 600 $;
   - `moyen` : s'ils sont compris strictement entre 600 $ et 1000 $;
   - `eleve` : s'ils sont supérieurs ou égaux à 1000 $.

   Pour cela, commencer par créer un vecteur `tuition.fees`, en répétant la modalité `moyen`, puis corriger (en deux temps) les modalités correspondant aux autres situations.
   Terminer en donnant le tableau des effectifs correspondant.
10. On souhaite ajouter au *data frame* les résultats de ces étudiants au *Base Year Composite Test Score*, un test évaluant le niveau de l'étudiant. Pour cela, importer les données du fichier bytest.txt et créer un nouveau *data frame* intitulé `studentbis` contenant les données de `student` et celle de `bytest`.
11. Construire un vecteur intitulé `commu` permettant d'identifier la communauté à laquelle appartient l'étudiant, en codant `Hispanic` pour *hispanic*, `Black` pour *black* et `White` pour ni *black*, ni *hispanic*. L'ajouter au *data frame*.
12. Pour chacune des communautés, déterminer la moyenne du *bytest* avec la commande suivante.

    ```
    by(studentbis$score,commu,mean)
    ```

13. En utilisant la commande `by` et son aide, renvoyer les quartiles des *bytest* par sexe.
14. On souhaite obtenir les écarts type des *bytest* par sexe. Cependant la commande `sd` renvoie l'écart type corrigé. On peut créer la fonction `et` calculant l'écart type avec le commande suivante.

    ```
    et<-function(x) {(length(x)-1)/length(x)*sd(x)}
    et(studentbis$score)
    ```

    L'utiliser pour renvoyer les écarts type des *bytest* par sexe.
15. Construire une nouvelle variable intitulée `par.sup` qui vaut 0 si aucun parent n'est diplômé du supérieur et 1 sinon. Renvoyer les déciles du *bytest* pour ces deux groupes.

Annexe

Exercices d'annales

Voici trois extraits de contrôle sur poste ou de session 2. Il n'y a et n'aura pas de correction. Par contre, vous avez bien sûr le droit de nous poser des questions et de nous montrer votre travail sur ces exercices d'annales pour voir si cela correspond bien à ce qui est attendu.

Exercice 18 (Extrait de l'examen session 2 - 2014-2015).
Le fichier de données se nomme `session2-1415.txt` et se trouve sur Celene. Il contient les données brutes d'un sondage réalisé auprès d'individus regardant quotidiennement la télévision. On a répertorié :

- le genre de l'individu `Genre`,
- le nombre de personnes dans le foyer `Foyer`,
- le diplôme obtenu `Diplome` (SupBac+2 signifie supérieur à Bac +2),
- la durée moyenne quotidienne devant la télévision, `Duree`, en minutes.

1. Importer les données dans un *data frame* intitulé `data`.
2. Déterminer la taille de l'échantillon.
3. Déterminer la structure de `data` et préciser les types des variables.
4. Donner un résumé des variables. Donner la liste des diplômes par une commande spécifique.
5. Effectuer l'histogramme représentant la durée quotidienne devant la télévision.
6. Calculer le nombre de personnes qui regardent 2h ou plus par jour en moyenne la télévision.
7. Extraire dans un *data frame* intitulé `dataF` les données concernant les femmes.
8. Supprimer la colonne `Genre` de `dataF`.
9. Effectuer les diagrammes en boîte de la durée quotidienne devant la TV en fonction du nombre de personnes dans le foyer. Soigner le graphique.
10. Déterminer la moyenne arrondi au dixième de la durée quotidienne devant la TV pour les individus appartenant à un foyer comportant 4 personnes.
11. Créer un vecteur intitulé `Vec` qui prend la valeur 1 si l'individu possède le bac ou un diplôme supérieur, et qui prend 0 sinon.
12. Calculer la proportion, arrondie au millième, d'hommes qui ont le bac ou plus dans l'échantillon .
13. Déterminer le tableau de contingence du couple de variables `Genre` et `Diplome`.

Exercice 19 (Session 1 - 2017-2018).
Vous allez travailler sur les données des employés et services d'une entreprise. Elles sont contenues dans les fichiers `company.csv` et `info_unit.csv` sur Celene. Ces tableaux de données contient les informations suivantes.
Dans le fichier `company.csv`, on a les informations suivantes concernant 14999 employés :

- satisfaction_level : niveau de satisfaction de l'employé ;
- last_evaluation : dernière évaluation ;
- number_project : nombre de projets ;
- average_monthly_hours : nombre moyen d'heures par mois ;
- time_spent_company : temps passé dans l'entreprise ;
- Work_accident : s'il a déjà eu un accident de travail ;
- left : si l'employé a quitté l'entreprise ;
- promotion_last_5years : s'il a eu une promotion au cours des 5 dernières années ;
- unit : département ;
- salary : niveau de salaire.

Dans le fichier `info_unit.csv`, on a des informations concernant les départements :

- unit : département ;
- effectif_serv : effectif du département ;
- nb_accident_serv : nombre de personnes du département ayant déjà eu des accidents de travail ;
- nb_left_serv : nombre de personnes du service ayant quitté l'entreprise.

1. Travail sur `info_unit.csv`.
   (a) Importer les données du fichier `info_unit.csv` dans un *data frame* intitulé `unit`.
   (b) Créer un vecteur contenant les taux de départ par département arrondis à 2 chiffres après la virgule.
   (c) Ajouter ce vecteur au *data frame* `unit` et en supprimer la variable nb_left_serv.
2. Travail sur `company.csv`.
   (a) Importer les données du fichier `company.csv` dans un *data frame* intitulé `comp`.
   (b) Déterminer le nombre d'employés du département IT ayant déjà eu un accident du travail.
   (c) Créer un document pdf intitulé Nom-graph.pdf comprenant un histogramme des niveaux de satisfaction des employés ayant quitté l'entreprise.
   (d) Représenter par des diagrammes en boîte la répartition des niveaux de satisfaction selon que l'employé a quitté ou non l'entreprise.
   (e) Ajouter sur le graphique une droite horizontale colorée au niveau du troisième quartile des niveaux de satisfaction globaux ainsi qu'un texte de la même couleur spécifiant qu'elle correspond au troisième quartile.
3. Créer le tableau de contingence des variables unit et salary. Ajouter les marges et renommer les titres des colonnes et ligne de marges.
4. Effectuer le tableau des proportions des différents départements arrondis à 3 chiffres après la virgule.
5. Modifier les *levels* de la variable unit afin de fusionner les départements dont les effectifs représentent moins de 7% des effectifs de l'entreprise en une modalité autres. Puis réaliser un graphique représentant les proportions des différents départements.

Exercice 20 (Session 2 -2016-2017).
Vous allez travailler sur des données concernant des maladies cardiaques. Elles sont contenues dans le fichier `maladiecardiaque_homme.csv` présent sur Celene. Ce tableau de données contient l

- age : l'âge du patient ;
- chest_pain : le type de douleur à la poitrine ;
- rest_bpress : la pression sanguine au repos ;
- blood_sugar : la présence d'un taux élevé de sucre dans le sang ;
- rest_electro : les anormalités détectées lors de la réalisation d'un électrocardiogramme au repos ;
- max_heart_rate : le rythme cardiaque maximal ;
- exercice_angina : la présence d'une angine de poitrine suite à un effort ;
- disease : présence d'une maladie cardiaque.

1. Importer les données dans un *data frame* intitulé `maladie`.
2. Représentations graphiques.
   (a) Représenter graphiquement les proportions des différents types de douleurs à la poitrine.
   (b) Représenter graphiquement le rythme cardiaque maximal en fonction de l'âge.
   (c) Réaliser un histogramme des rythmes cardiaques maximums pour les patients ayant eu une angine de poitrine et un autre pour ceux n'en ayant pas eu.
   Veiller à réaliser ces histogrammes de façon à ce qu'ils soient facilement comparable (échelles).
3. Étude numérique des données.
   (a) Quel est le rythme cardiaque moyen des personnes malades ?
   (b) Donner en une unique commande les différents quartiles de la pression sanguine des personnes non malades.
   (c) Déterminer le nombre de patients ayant un taux de sucre dans le sang élevé.
4. Créer la variable high_heart_rate valant oui si le rythme cardiaque maximal est supérieur à 150 et non sinon.
5. Extraire dans un *data frame* intitulé malade les données des individus malades (sans la variable disease).