INSA

Année universitaire 2018-2019

3ème année INFO BD & MI

STATISTIQUES

TP 1 : Introduction au logiciel R

R est un logiciel de statistique distribué gratuitement par le CRAN (Comprehensive R Archive Network) à l'adresse suivante `http://cran.r-project.org`. L'installation de R est très simple.

1 La session de travail

Lancer R à partir du menu *Démarrer*. R ouvre une fenêtre avec un certain nombre de commentaires. A la fin de cet affichage apparaît l'invite de commande > qui indique que R attend maintenant une instruction. Chaque instruction doit être validée par `Entrée` pour être exécutée.

- Si l'instruction est correcte, R redonne la main avec l'invite de commande.
- Si l'instruction est incomplète, R retourne le signe +. Il faut compléter l'instruction ou sortir de cette situation en tapant sur `Echap`.
- Si l'instruction est erronée, un message d'erreur apparaît.

Vous pouvez, par exemple, demander à R de faire des calculs.

```
> 5*(-3.2) # Attention, le séparateur décimal doit est un point.
> 5^2
> cos(4*pi/3)
```

Tout code R qui suit le caractère # est considéré par R comme un commentaire. En fait, il n'est pas interprété par R.

Pour connaître le répertoire de travail courant de R, taper **getwd()** dans la console. Il est possible de modifier le répertoire de travail avec la fonction **setwd()** ou en allant dans `Fichier/Changer le répertoire courant`.

Pour quitter une session R, on utilise la fonction **quit**.

```
> q()
```

Il vous est alors proposé de sauver une image de la session. Si vous répondez oui, un fichier ayant pour extension `.Rdata` sera créé dans votre répertoire de travail. En cliquant sur ce fichier, une nouvelle session s'ouvrira directement dans le répertoire concerné. Les objets créés à la session précédente seront de nouveau accessibles.

Lors de chaque session, **il est fortement recommandé de taper vos instructions R dans une fenêtre de script** appelée *script* ou *R Editor*, accessible depuis le menu `Fichier/Nouveau script`. Vous pouvez sauvegarder ce script dans votre répertoire de travail, sous le nom `monscript.R` par exemple, en cliquant dans le menu `Fichier/Sauver sous` et le rouvrir lors d'une session ultérieure depuis le menu `Fichier/Ouvrir un script`. Il est préférable que le nom du script et de son chemin d'accès ne comprennent **pas de caractères accentués**.

Dans un script, les instructions sont, en général, saisies les unes sous les autres. Vous pouvez taper successivement les combinaisons de touches `CTRL+A` pour sélectionner l'ensemble de ces instructions,

CTRL+R pour les coller et les exécuter en une seul étape dans la console R. Vous pouvez aussi exécuter une seule ligne d'instructions R du script en tapant CTRL+R lorsque le curseur clignotant se trouve sur la ligne en question dans la fenêtre de script.

Vous pouvez également utiliser la fonction **source()** depuis la console R pour aller lire et exécuter l'intégralité du contenu de votre script. Cela évite de surcharger inutilement la console.

Pour obtenir de l'aide sur une fonction, par exemple la fonction **mean**, il suffit d'utiliser une des deux instructions suivantes :

```
> help(mean)
> ?mean
```

Vous pouvez également être amenés à utiliser des packages. Un **package** (paquet ou bibliothèque de programmes externes) est simplement un ensemble de programmes R qui complète et permet d'augmenter les fonctionnalités de R. Un package est généralement dévolu à des méthodes particulières ou à un domaine d'applications. Il existe un très grand nombre de packages. Certains considérés comme indispensables sont fournis avec R. Les autres peuvent être téléchargés librement sur le réseau CRAN.

Pour installer un package, il faut utiliser la fonction **install.packages** ou cliquer sur `Packages/Install package(s)...`, puis choisir la miroir le plus proche de vous et sélectionner le package à installer. Il reste alors à le charger pour pouvoir l'utiliser dans une session R grâce à l'instruction `library(nom_package)`.

2 Les objets R

2.1 Création, affichage et suppression d'un objet

```
> b<-25.2
> b
> x=b
> x
> y="mot"
> mode(x)
> mode(y)
> str(x)
> str(y)
> ls()
> rm(b)
> ls()
>
> x=NA
> print(x+1)
> is.na(x)
```

2.2 Les vecteurs

Le vecteur est un objet composé d'un ensemble de valeurs appelées composantes ou éléments. Tous les éléments d'un vecteur sont du même type.

2.2.1 Créations de vecteurs

```
> # Vecteurs numériques
> x<-c(5.6,-2,78,45.8)
> x
```

```
> 1:6
> seq(from=1,to=6,by=0.5)
> rep(1,4)
> rep(c(1,2),each=3)
>
> # Vecteurs de caractères
> x<-c("BD","MI")
> x
>
> # Vecteurs de logiques
> 1>0
> x<-c(-1,0,2)
> x==0
> x!=0
> x>1
> (1+x^2)*(x>1)
```

2.2.2 Fonctions utiles pour les vecteurs

```
> x<-c(1,3,6,2,7,4,8,1,-1)
> length(x)
> sort(x)
> which.min(x)
> which.max(x)
> sum(x)
> prod(x)
> mean(x)
> sd(x) # Ecart-type biaisé
```

2.2.3 Calcul vectoriel

```
> x<- c(1,2,4,6,3)
> 3*x
> x^2
> log(x)
>
> y<- c(4,7,8,1,1)
> x+y
> x*y
> t(x)%*%y
>
> #Recyclage
> u<-1:15
> v<-1:5
> u+v
```

2.2.4 Extraction d'éléments d'un vecteur

```
> # Extraction par indice
> vec<-c(2,4,6,8,3)
> vec[2]
> vec[-2]
> vec[2:5]
> vec[-c(1,5)]
>
> # Extraction par masque logique
> vec[c(T,F,F,T,T)]
> vec>4
> vec[vec>4]
> x<-1:5
> y=c(-1,2,-3,4,-2)
> x[y>0]
```

2.2.5 Insertion d'éléments dans un vecteur

```
> z<-c(0,0,0,2,0)
> z[1]=4
> z[c(1,5)]<-1
> z[which.max(z)]<-0
> z
> z[z==0]<-8
> z
```

2.3 Les facteurs

Les facteurs sont des vecteurs permettant la manipulation de données qualitatives. Ils forment une classe d'objets particulière et bénéficient de traitements particuliers pour certaines fonctions telles que les graphiques.

2.3.1 Création de facteurs

```
> sexe<-factor(c("M","M","F","M","F","M","M","M"))
> sexe
> sexe<-factor(c(2,2,1,2,1,2,1), labels=c("femme", "homme"))
> sexe
>
> x=c(1:5,5:1)
> x
> x.f<-as.factor(x)
> x.f
```

2.3.2 Fonctions utiles pour les facteurs

```
> levels(x.f)
> nlevels(x.f)
> table(x.f)
```

2.4 Les matrices

Tous les éléments d'une matrice sont de même type. Chaque élément de la matrice peut être repéré par son numéro de ligne et son numéro de colonne.

2.4.1 Création de matrices

```
> M= matrix(1:8,nrow=2)
> M
> matrix(1:8,nrow=2,byrow=TRUE)
>
> diag(5)
> diag(1:4)
>
> cbind(1:4,5:8)
> rbind(1:4,5:8)
>
> x<-seq(1,10,by=2)
> is.matrix(x)
> as.matrix(x)
```

2.4.2 Fonctions utiles sur les matrices

```
> M=matrix(1:6,nrow=2)
> dim(M)
> nrow(M)
> ncol(M)
> colnames(M)=c("var1", "var2","var3")
> rownames(M)=c("individu1","individu2")
> M
```

Ces fonctions peuvent également être utilisées sur les dataframes.

2.4.3 Calcul matriciel

```
> A=matrix(1:4,ncol=2)
> A
> exp(A)
> A^3
> t(A)
> diag(A)
> det(A)
> solve(A)
>
> B=matrix(c(5,7,6,8),ncol=2)
> B
> A+B
> A*B
> A%*%B
>
> # Recyclage
```

```
> matrix(1:4,ncol=3,nrow=3)
>
> # Fonction apply
> X<-matrix(c(1:4,1,6:8),nrow=2)
> X
> apply(X, MARGIN=1, FUN=mean)
> apply(X, MARGIN=2, FUN=sum)
```

2.4.4 Extraction d'éléments d'une matrice

```
> # Extraction par indice
> M=matrix(1:12,nrow=4,ncol=3,byrow=TRUE)
> M
> M[2,3]
> M[,1]
> M[c(1,4),]
> M[3,-c(1,3)]
>
> ind=c(2,4,6,8,3)
> M[ind]
>
> # Extraction par masque logique
> Mlogique=matrix(c(TRUE,FALSE),nrow=4,ncol=3)
> Mlogique
> M[Mlogique]
> M[M>2]
```

2.4.5 Insertion d'éléments dans une matrice

```
> m=matrix(c(1,2,3,1,2,3,2,1,3),3,3)
> m[1,3]=4
> m[m!=2]<-0
> m
> M<-M[-4,]
> M
> m[M>7]<-M[M>7]
> m
```

2.5 Les listes

La liste est un objet hétérogène. C'est un ensemble ordonné d'objets qui n'ont pas toujours le même mode ou la même longueur. Les objets sont appelés composantes de la liste.

2.5.1 Création de listes

```
> vecteur<-seq(2,10,by=3)
> matrice<-matrix(1:8,ncol=2)
> facteur=factor(c("M","M","F","M","F","F","M"))
> maliste<-list(vec=vecteur, mat=matrice, fac=facteur)
```

```
> str(maliste)
> length(maliste)
> mode(maliste)
> names(maliste)
```

2.5.2 Extraction d'éléments d'une liste

```
> L<-list(12,c(34,67,8),M,1:15,list(a=10,b=11))
> L
> L[[2]]
> L[[2]][2]
> L[[5]][[2]]
>
> L<-list(voitures=c("ford","toyota", "BMW"), climat=c("tropical","tempéré"))
> L[["voitures"]]
> L$voitures
```

Comme pour les data-frames, il est possible d'attacher une liste grâce à la fonction `attach()` pour avoir accès directement à ses élements en tapant leurs noms dans la console.

2.5.3 Insertion d'éléments dans une liste

```
> L$climat[2]="Continental"
> L
```

2.6 Les dataframes

Les data-frames sont des listes particulières dont les composantes sont de même longueur, mais les modes peuvent être différents. Les tableaux de données usuels utilisés en statistique sont souvent considérés comme des data-frames.

2.6.1 Création de dataframes

```
> vec1<-1:5
> vec2=c("a","a","c","v","b")
> df<-data.frame(nom.var1=vec1, nom.var2=vec2)
> df
> str(df)
```

2.6.2 Extraction/insertion dans les data-frames

L'extraction et l'insertion dans les dataframes fonctionnent de la même façon que dans les matrices.

```
> tab=data.frame(nom=c("Martin","Durand","Dupont"),
+                prenom=c("Gilles", "Luc", "Laura"),age=c(45,35,37))
> tab
> tab$prenom
>
> prenom
```

```
> attach(tab)
> prenom
> age
> detach(tab)
> age
```

2.7 Les fonctions

Une fonction est un objet R. Un grand nombre de fonctions sont prédéfinies dans R, mais il est possible de créer ses propres fonctions. Une fonction admet des arguments en entrée et retourne un résultat en sortie. Les arguments d'une fonction sont soit obligatoires soit optionnels. Dans ce dernier cas, ils possèdent une valeur par défaut.

Le programmation et l'utilisation de fonctions R sont similaires à celles des fonctions MATLAB. Seule la syntaxe est légèrement différente.

2.7.1 Exemples de fonctions

```
> bonjour<-function(nom="Pierre") cat("Bonjour",nom,"!\n")
> bonjour
> bonjour()
> bonjour("Sabrina")
>
> racine=function(a,b,c)
+ {
+   # Fonction qui calcule le discriminant et les racines réelles d'un trinôme
+   # Arguments : coefficients du trinôme ax^2+bx+c
+   discriminant=b^2-4*a*c
+   rac=NULL
+   if(discriminant==0)
+     rac=-b/(2*a)
+   if(discriminant>0)
+     rac=c((-b-sqrt(discriminant))/(2*a),(-b+sqrt(discriminant))/(2*a))
+   return(list(discriminant=discriminant,racines= rac))
+ }
>
> racine
> racine(1,1,1)
> racine(-2,-1,1)
```

2.7.2 Structures de contrôle utiles pour écrire des fonctions

- **Instructions `if` et `else`**
  L'instruction conditionnelle **if** est utilisée sous les deux formes suivantes : `if(<cond>)<expr:vrai>` ou `if(<cond>)<expr:vrai> else <expr:faux>`.
  La paramètre `<cond>` doit être un logique qui prend l'une des valeurs `TRUE` or `FALSE`.

  ```
  > x<-2
  > y<-3
  > if(x<=y)
  +   {
  +     z<-y-x
  +     print("x plus petit que y")
  ```

```
+   } else
+   {
+     z<-x-y
+     print("x plus grand que Y")
+     z
+   }
```

```
> # Attention
> x<-0.1
> y<-0.1
> x==y
> x<-0.2-0.1
> y<-0.3-0.2
> x==y
> all.equal(x,y)
```

- **Instruction `for` (pour)**
  La syntaxe de cette instruction est la suivante : `for(i in vec) <Instructions>`.

```
> for(i in 1:3) print(i)
> x<-c(1,3,7,2)
> for(var in x)
+ {
+   print(var-10)
+   print(2*var)
+ }
```

- **Instruction `while` (tant que)**
  La syntaxe de cette instruction est la suivante : `while(<condition>)<expression>`.

```
> x<-2
> y<-1
> while(x+y<7)  x<-x+y
> x
```

- **Instruction `repeat` (répéter)**

```
> i<-0
> repeat
+ {
+   i<-i+1
+         if(i==4) break
+ }
```

3 Importer et générer des données dans R

3.1 Importer des données depuis un fichier `.txt` ou `.csv`

```
> # ATTENTION, votre répertoire de travail doit contenir le fichier à importer.
> donnees<-read.table(file="nom_fichier.txt")
> head(donnees)
> summary(donnees)
>
> read.csv("nom_fichier.csv")  # Fichier .csv + Données séparées par des virgules
> read.csv2("nom_fichier.csv")  #Fichier .csv + Données séparées par des points-virgules
>
> temperature=read.table("http://www.biostatisticien.eu/springeR/temperature.dat")
```

Les principaux paramètres de la fonction `read.table()` sont les suivants :

- `file="chemin/fichier"` : Emplacement et nom du fichier à lire. Si le paramètre prend la valeur `file.choose()` une fenêtre de dialogue s'ouvrira pour sélectionner le fichier.
- `header=TRUE` : Valeur logique indiquant si le fichier contient des noms de variables sur la première ligne
- `sep="\t"` : Séparateur de champs
- `dec="."` : Séparateur décimal pour les nombres
- `row.names=1` : Indique si la première colonne contient le nom des individus. Si ce n'est pas le cas, il faut omettre ce paramètre.

3.2 Générer des données aléatoirement

Il existe de nombreuses fonctions permettant de tirer des nombres aléatoirement. En voici quelques exemples :

```
> sample(1:8,3)
> sample(1:8,6,replace=TRUE)
> runif(3)
> runif(5,min=-2,max=5)
> rnorm(4, mean=2, sd=3)
> rt(10,4)
> rchisq(6,5)
> rf(5,4,3)
```

Le logiciel R permet de déterminer les quantiles associés à ces lois de probabilité.

```
> qnorm(0.975)
> qnorm(0.975, mean=2, sd=3)
> qt(0.975,4)
> qf(0.95,4,3)
```

4 Graphiques

```
# Fonctions de tracé de bas niveau
plot(1:4,c(2,3,4,1),type="b",main="Graphique", xlab="x", ylab="y", ylim=c(0,5),
     col="blue")
points(1:4,c(4,2,1,3),type="l", col=2)
abline(h=1.5)
abline(v=3, lty=2)
abline(a=5.5,b=-2, col=3)
text(3.1,4.5,"x=3")

curve(sin(x),from=-pi, to=pi, col=2)
curve(cos(x),from=-pi, to=pi, col=3,add=TRUE)
legend(x=-3,y=1,legend=c("sinus","cosinus"), col=2:3,lty=1)
```

```
# Représentation d'une série statistique
x=rnorm(10000)
hist(x)
boxplot(x)
y=rpois(100,5)
barplot(table(y))
```

Exercices

Exercice 1 : Création et manipulation de vecteurs

1. Créer les 3 vecteurs ci-dessous à l'aide de la fonction **rep**.
   ```
   (1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
   (1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5)
   (1, 1, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4)
   ```
2. Donner une instruction permettant de remplacer les valeurs manquantes d'un vecteur `x` quelconque par des zéros.
3. Donner une instruction permettant de remplacer les éléments à valeur négative d'un vecteur de numérique `y` par leur opposé.
4. Donner une ligne de code permettant de calculer la somme suivante : $\sum_{i=10}^{100}(i^3+4i^2)$.
5. Soient $\mathbf{x}=(x_1,x_2,\ldots,x_n)$ et $\mathbf{y}=(y_1,y_2,\ldots,y_n)$ deux vecteurs de taille n.
   (Par exemple, vous pouvez prendre `x=rnorm(100)` et `y=sample(0:99,100,replace=TRUE)`.)
   (a) Créer le vecteur $(y_2-x_1,\ldots,y_n-x_{n-1})$.
   (b) Créer le vecteur $\left(\frac{\sin(y_1)}{\cos(x_2)},\frac{\sin(y_2)}{\cos(x_3)},\ldots,\frac{\sin(y_{n-1})}{\cos(x_n)}\right)$.
6. Fixer la graine du générateur de nombres aléatoires (**set.seed**) à la valeur 007 et créer un vecteur `vec1` de 100 nombres uniformément tirés entre 0 et 7 (**runif**). Calculer la moyenne empirique (**mean**) et la variance empirique des valeurs contenues dans `vec1`.

Exercice 2 : Création manipulation de matrices

1. Créer la matrice `mat` suivante avec les noms de lignes et de colonnes :

	colonne 1	colonne 2	colonne 3	colonne 4
ligne-1	1	5	5	0
ligne-2	0	5	6	1
ligne-3	3	0	3	3
ligne-4	4	4	4	2

2. Créer un vecteur contenant les éléments diagonaux de la matrice `mat`.
3. Créer une matrice contenant uniquement les deux premières lignes de la matrice `mat`.
4. Créer une matrice contenant uniquement les deux dernières colonnes de la matrice `mat`.
5. Créer une matrice contenant toutes les colonnes de la matrice `mat` sauf la troisième.
6. Calculer le déterminant puis inverser la matrice `mat` en utilisant les fonctions appropriées.
7. Créer un vecteur contenant les moyennes des valeurs de chaque colonne de la matrice `mat`.

Exercice 3 : Calcul matriciel-Régression multiple

Considérons l'ensemble des données enregistrées comme data-frame dans R sous le nom `mtcars`. Cette base de données contient les mesures de 11 variables liées à l'aspect et aux performances d'un ensemble de 32 voitures (modèles 1973-1974). On considère le modèle suivant :

$$\text{mpg}=\beta_0+\beta_1\text{hp}+\beta_2\text{wt}+\varepsilon$$

où `mpg` est un vecteur donnant la consommation de carburant du véhicule (gallons par miles), `hp` un vecteur donnant sa puissance (nombre de chevaux) et `wt` un vecteur donnant son poids (milliers de livres). Le vecteur ε est un vecteur d'erreurs dont les termes sont indépendants et identiquement distribués suivant une loi normale $\mathcal{N}(0,\sigma^2)$.

On cherche à estimer les paramètres β_0, β_1 et β_2 par la méthode des moindres carrés ordinaires.

1. A partir de `mtcars`, créer le vecteur $\mathbf{y}$ contenant les valeurs de consommation de carburant (`mpg`).
2. Créer une matrice $\mathbf{X}$ qui aura autant de lignes que $\mathbf{y}$ et 3 colonnes. La première colonne ne contiendra que des 1, la seconde contiendra le vecteur `hp` et la troisième le vecteur `wt` (On pourra utiliser la fonction **cbind**).
3. Calculer la transposée de $\mathbf{X}$: $\mathbf{X}^T$.
4. Calculer de deux façons différentes le produit matriciel $\mathbf{X}^T\mathbf{X}$.
5. Calculer l'inverse de ce produit matriciel : $(\mathbf{X}^T\mathbf{X})^{-1}$.
6. Calculer le produit matriciel $\mathbf{X}^T\mathbf{y}$.
7. Donner les estimations des paramètres $\widehat{\beta}_0$, $\widehat{\beta}_1$, $\widehat{\beta}_2$, c'est-à-dire calculer $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$.
   Vérifier votre résultat en saisissant l'instruction `lm(mpg~1+hp+wt,data=mtcars)` dans la console.

Exercice 4 : Importation de données et fusion de tables

1. Importer les fichiers `test1.csv`, `test2.csv` et `test3.csv` après avoir examiné leur contenu.
2. (a) Importer les fichiers `etat1.csv`, `etat2.csv` et `etat3.csv`.
   (b) Fusionner les trois data-frames en un seul en utilisant les colonnes communes afin de répéter les lignes de manière adéquate.

Exercice 5 : Comparaison de distributions

1. A l'aide de la fonction **curve**, tracer la densité de la loi normale $\mathcal{N}(0,1)$ entre -4 et 4 (utiliser **dnorm**). Ajouter le titre suivant : `Comparaison de distributions`.
2. Tracer sur le même graphique la densité des lois de Student à 5 et 30 degrés de liberté. Utiliser la fonction **curve** et une couleur différente pour chaque courbe.
3. Reprendre les questions 1 et 2 en utilisant respectivement les fonctions **plot** et **lines**.
4. Ajouter une légende en haut à gauche pour différencier chaque distribution.

Exercice 6 : Tracé de courbes

1. Représenter sur un même graphique sur l'intervalle $[-3;3]$:
   - la fonction $f(x) = x^2 + 1$ en rouge,
   - la droite d'équation $x = 0$ en bleu,
   - la fonction $g(x) = 2x + 2$ en vert,
   - la fonction $h(x) = \begin{cases} x^2 + 2x + 3 & \text{si } x < 0 \\ x + 3 & \text{si } 0 \leq x \leq 2 \\ x^2 + 4x - 7 & \text{si } x > 2 \end{cases}$ en noir.
2. Ajouter une légende au graphique.

Exercice 7 : Harvey, Irma, Jose, Maria et les autres...

Depuis 1953, les cyclones se développant dans l'océan Atlantique nord sont identifiés par des prénoms. Un même prénom peut éventuellement être utilisé plusieurs fois. Cependant, les pays affectés par des cyclones particulièrement intenses ayant causé de forts dommages peuvent demander à l'Organisation Météorologique Mondiale de retirer le nom de ceux-ci des listes futures. Il existe ainsi une liste de noms retirés de cyclones. Ce sont derniers qui vont faire l'objet de notre étude. Pour cela, vous disposez de 3 fichiers de données.

- Le fichier `Intensite.txt` comprend 5 variables `nom`, `annee`, `noeuds`, `km/h` et `pression`.
- Le fichier `Dommages.txt` comprend 3 variables `nom`, `annee` et `cout`.
- Le fichier `Mortalite.csv` comprend 4 variables : `nom`, `annee`, `morts` et `lieu`.

Le descriptif des variables est donné ci-dessous :

`nom`	nom du cyclone
`annee`	année où a eu lieu le cyclone
`noeuds`	mesure des vents maximaux soutenus sur une minute en noeuds
`km/h`	mesure des vents maximaux soutenus sur une minute en km par heure
`pression`	pression centrale minimale en hPa
`cout`	coût des dommages normalisé en milliards de dollars américains de 2015
`morts`	nombre de morts dûs au cyclone
`lieux`	lieux frappés par le cyclone

Les cyclones se formant dans l'hémisphère ouest sont classés suivant leur intensité selon l'échelle de Saffir-Simpson qui compte 7 catégories :

Vitesse maximale des vents sur une minute	**Catégorie**
entre 0 et 33 noeuds	dépression
entre 34 et 63 noeuds	tempête
entre 64 et 82 noeuds	ouragan de catégorie 1
entre 83 et 95 noeuds	ouragan de catégorie 2
entre 96 et 113 noeuds	ouragan de catégorie 3
entre 114 et 135 noeuds	ouragan de catégorie 4
strictement supérieure à 135 noeuds	ouragan de catégorie 5

1. Lire les 3 fichiers `Intensite.txt`, `Dommages.txt` et `Mortalite.csv` et stocker leur contenu dans des data-frames nommés respectivement `intensite`, `dommages` et `mortalite`.
2. Supprimer du tableau `intensite` les lignes qui correspondent à des dépressions et des tempêtes, c'est-à-dire les observations pour lesquelles la mesure des vents maximaux soutenus sur une minute est strictement inférieure à 64 noeuds.
3. Supprimer du tableau `dommages` les lignes contenant des données manquantes.
4. Créer un facteur `categorie` donnant la catégorie de chaque ouragan et ajouter cette variable au tableau `intensite`.
5. Fusionner les trois tableaux `intensite`, `dommages`, `mortalite` dans un nouveau tableau nommé `ouragan`.
6. Trier le data-frame `ouragan` par ordre chronologique de la survenue des ouragans.
7. Calculer le coût moyen des dommages pour chaque catégorie d'ouragans.
8. Tracer le diagramme en bâtons du nombre d'ouragans par année.