Initiation à R

Allou Samé
allou.same@ifsttar.fr

2017/2018

Notion d'objet

- R est un langage orienté objet : tous les éléments manipulés dans R sont qualifiés d'objets.
- Chaque objet est caractérisé par un nom, une classe et des attributs.
- Exemple de classes d'objets : vecteur, matrice, tableau, liste, ...
- L'utilisateur peut effectuer des actions sur ces objets via des fonctions.

Opérations arithmétiques

- Opérations élémentaires

```
> 1 + 2 + 3 # addition
[1] 6

> 3 * 4 + 2 # addition et multiplication
[1] 14

> 1 + 3/2 # addition et division
[1] 2.5

> (1 + 3)/2 # ne pas hésiter à utiliser des parenthèses
[1] 2

> 4^2 # puissance
[1] 16
```

- Le symbole # permet d'insérer des commentaires pour mieux expliquer un programme. Toute commande située après ce symbole n'est pas exécutée.

Fonctions numériques

- Racine carrée

```
> sqrt(4)
[1] 2
```

- Logarithme

```
> log(pi)
[1] 1.14473
```

- Autres fonctions

```
abs()
sign()
exp()
sin(), asin()
cos(), acos()
tan(), atan()
```

Assigner des valeurs à des objets

- On peut assigner des valeurs à des objets en utilisant les opérateurs = ou <-
- Exemple :
  - assignons la valeur `5` à la variable `x`

```
> x <- 5
> x
[1] 5
```

  - on peut maintenant effectuer des calculs et définir d'autres variables à partir de la variable `x`

```
> y <- sqrt(x + 4)
> y
[1] 3
```

Vecteurs (1/2)

- Vecteur de valeurs numériques

```
> c(2, 3, 5, 2, 7, 1)
[1] 2 3 5 2 7 1
```

- Vecteur de valeurs logiques

```
> c(T, F, F, F, T, T, F)
[1] TRUE FALSE FALSE FALSE TRUE TRUE FALSE
```

- Vecteur de chaînes de caractères

```
> c("Bruxelles", "Paris", "Canberra", "Sydney")
[1] "Bruxelles" "Paris" "Canberra" "Sydney"
```

Vecteurs (2/2)

- On peut affecter un vecteur à une variable

```
> x <- c(2,4,6,8)
> x
[1] 2 4 6 8
```

- Les fonctions usuelles appliquées à un vecteur s'appliquent à chaque élément de ce vecteur

```
> x^2
[1]  4 16 36 64
> log(x)
[1] 0.6931472 1.3862944 1.7917595 2.0794415
```

- Les opérations usuelles appliquées à deux vecteurs de même taille sont effectuées élément par élément

```
> y<-c(1,2,3,4)
> x+y
[1]  3  6  9 12
> x*y
[1]  2  8 18 32
```

- Produit scalaire de deux vecteurs

```
> x %*% y
     [,1]
[1,]   60
```

Création de vecteurs particuliers

- Séquence de nombres

```
> x <- seq(10,50,by=1)
> x
 [1] 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34

> x=10:50
> x
 [1] 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
```

- Dupliquer des nombres

```
> rep(3,4)
[1] 3 3 3 3
```

- Taper les commandes suivantes et observer les résultats

```
> seq(1,10,2)
> seq(5,1)

> x<-c(1,2,3)
> rep(x,3)

> y<-x
> rep(x,y)
```

Manipulation de vecteurs (1/2)

- Extraction d'élément

```
> x<-1:5
> x[3]   # 3ème élément du vecteur x
 [1] 3

> x[-3]  # x privé de son 3ème élément
 [1] 1 2 4 5
```

- Remplacement d'élément

```
> x[3] <- 10
> x
 [1] 1 2 10 4 5
```

- Ajout d'élément

```
> x[6] <- 6
> x
 [1] 1 2 10 4 5 6

> x <- c(x,7)
> x
 [1] 1 2 3 10 4 5 6 7
```

Manipulation de vecteurs (2/2)

- Utilisation d'opérateurs relationnels <, <=, >, >=, ==, !=

```
> x <- c(3, 11, 8, 15, 12)
> x > 8
[1] FALSE TRUE FALSE TRUE TRUE
> x != 8
[1] TRUE TRUE FALSE TRUE TRUE
```

- Extraction d'éléments

```
> x <- c(3, 11, 8, 15, 12)
> x[c(2,4)]
[1] 11 15
```

- Extraction d'éléments à l'aide d'opérateurs relationnels

```
> x[x > 10]
[1] 11 15 12
```

Chaînes de caractères

- Chaîne de caractère

```
> x="apprentissage"
```

- Vecteur de chaînes de caractères

```
> x=c("Paris","Lyon","Marseille")
```

- Concaténation de chaînes de caractères

```
> x=paste("Apprentissage","non","supervisé")
> x=paste("Apprentissage","non","supervisé",sep="-")
> x=paste("X",1:10,sep="")
```

Données manquantes

Dans un jeu de données sous R, les données manquantes sont généralement désignées par la valeur `NA`

- Créer un vecteur contenant des valeurs manquantes

```
> x = c(1,3,5,NA,9,NA)
```

- Trouver les valeurs manquantes de x

```
> ind = is.na(x)
```

Fonctions applicables à des vecteurs

- Exemple de fonctions de vecteurs

```
> length(x) # Nombre d'éléments du vecteur x

> max(x)    # Plus grande valeur de x

> min(x)    # Plus petite valeur de x

> sum(x)    # Somme des éléments de x

> prod(x)   # Produit des éléments de x

> mean(x)   # Moyenne des éléments de x

> sd(x)     # Ecart-type des éléments de x

> var(x)    # Variance des éléments de x
```

Matrices

- Création de matrice
  - Première méthode

```
> x = matrix(1:6,2,3)
> x
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

  - Seconde méthode

```
> x = seq(1:9)
> dim(x) = c(3,3)
> x
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

  - Agrégation de vecteurs

```
> cbind(1:4,5:8,9:12)
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

```
> rbind(1:4,5:8,9:12)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
```

Data Frame ou tableau de données (1/3)

- Contrairement à une matrice, un tableau de données peut contenir plusieurs types de données (numériques, caractères, logiques, ...).
- Les vecteurs inclus dans le tableau doivent être de même longueur.
- Création d'un tableau de données :

```
> etudiants = data.frame( nom = c("Alfred","Paul",
+ "Isabelle","Mathieu"),  age = c(21,26,23,20) ,
+ sexe = c(rep("M",2),"F","M" ) )
> etudiants
       nom age sexe
1   Alfred  21    M
2     Paul  26    M
3 Isabelle  23    F
4  Mathieu  20    M
```

Data Frames ou tableaux de données (2/3)

- Autre manière de créer un tableau de données :

```
> nom <- c("Alfred","Paul","Isabelle","Mathieu")
> age <- c(21,26,23,20)
> sexe = c(rep("M",2),"F","M" )
> etudiants <- data.frame(nom,age,sexe)
> etudiants
       nom age sexe
1   Alfred  21    M
2     Paul  26    M
3 Isabelle  23    F
4  Mathieu  20    M
```

Indexation des éléments d'un tableau de données (3/3)

```
> etudiants
       nom age sexe
1   Alfred  21    M
2     Paul  26    M
3 Isabelle  23    F
4  Mathieu  20    M

> etudiants[2,]
   nom age sexe
2 Paul  26    M

> etudiants[etudiants$age>22,]
       nom age sexe
2     Paul  26    M
3 Isabelle  23    F
```

Listes

- Une liste est créée de la même façon qu'un tableau de données avec la fonction `list`.
- Il n'y a aucune contrainte sur la longueur des objets qui y sont inclus.
- A la différence de `data.frame()`, les noms des objets ne sont pas utilisés par défaut
- Exemples

```
> maliste1 <- list(nom,age,sexe)
> maliste1
[[1]] [1] "Alfred" "Paul" "Isabelle" "mathieu"
[[2]] [1] 21 26 23 20
[[3]] [1] "M" "M" "F" "M"

> maliste2 <- list(A=nom,B=age,C=sexe)
> maliste2
$A [1] "Alfred" "Paul" "Isabelle" "mathieu"
$B [1] 21 26 23 20
$C [1] "M" "M" "F" "M"
```

Indexation des éléments d'une liste

```
> maliste2
$A [1] "Alfred"   "Paul"     "Isabelle" "Mathieu"
$B [1] 21 26 23 20
$C [1] "M" "M" "F" "M"

> maliste2$A
[1] "Alfred"   "Paul"     "Isabelle" "Mathieu"

> malistea <- maliste2[2] # toute sous liste d'une liste est une liste
> malistea
$B [1] 21 26 23 20

> a <- maliste2[[2]]
> a
[1] 21 26 23 20
```

Importer des données dans R

- Définir un répertoire de travail dans R

```
> setwd("c:/mon_rep_R")
> getwd()
[1] "c:/mon_rep_R"
```

- Il est parfois utile d'importer des données dans R afin d'effectuer des calculs statistiques à partir de ces données. Ceci peut être effectué à l'aide des commandes suivantes :

```
> data = read.table("h:/data.dat") # manière basique
> data = read.table("h:/data.dat", header=T) # avec noms de colonnes
```

- Pour mieux prendre en compte les caractères de séparation des variables, on peut utiliser les commandes :

```
> data = read.table("h:/data.dat", header=T, sep=",")
> data = read.table("h:/data.dat", header=T, sep="\t")
```

Graphiques

- Ouvrir une nouvelle fenêtre graphique

```
> x11()
```

- Utilisation basique de la fonction `plot`

```
> x <- 1:20

> plot(x, exp(x), type="l", main="Main Title of Plot",
         xlab="titre axe abscisses", ylab="titre axe ordonnées")
```

- Options pour les titres

```
> xlab="titre axe abscisses"

> ylab="titre axe ordonnées"

> main="Main Title of Plot"
```

Graphiques : quelques options de la fonction “plot"

`type=`	Contrôle le type de graphique : (“l" : ligne, “p" : point, “b"=ligne et point)
`add=`	FALSE par défaut et TRUE si l’on souhaite superposer le graphe à celui existant
`xlim=, ylim=`	limites inférieures et supérieures des axes
`axes=`	TRUE ou FALSE pour afficher ou ne pas afficher les axes
`col=`	Contrôle de la couleur des symboles
`lty=`	Contrôle le type de ligne tracée (1 : continue, 2 : tirets, 3 : points, 4 : points et tirets alternés, 5 : tirets longs, 6 : tirets courts et longs alternés)
`pch=`	controle le type de symbole utilisé : entier entre 1 et 25
`xaxt` et `yaxt`	Contrôle l’affichage des axes des abscisses et des or

Graphiques : ajouts de lignes et de points

- Ajout de points au graphique courant

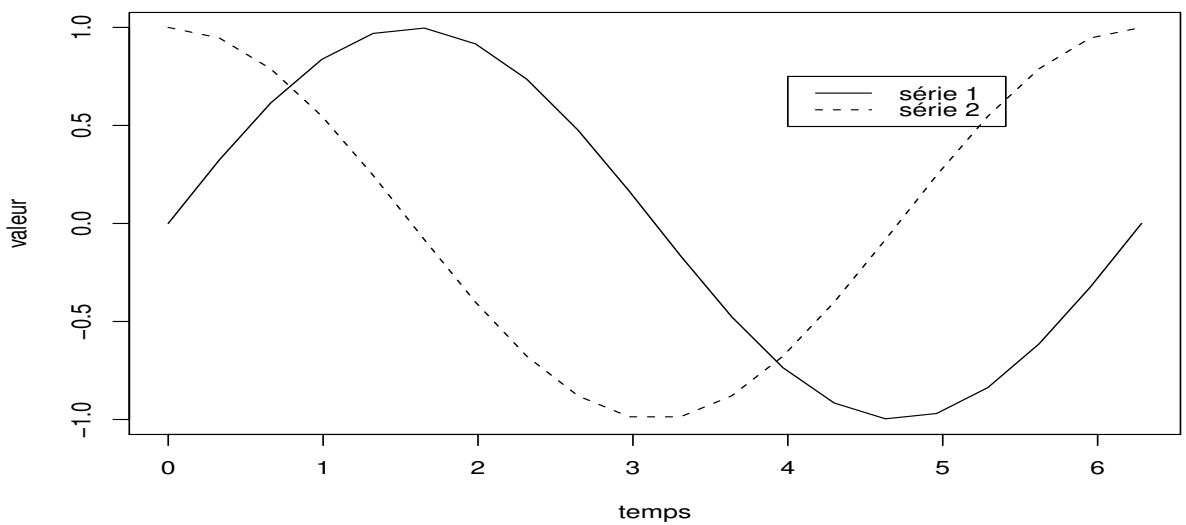
```
> points(x,y)
```

- Ajout de lignes au graphique courant

```
> lines(x,y,lty=2)
```

Ajouter une légende à un graphique

```
> ti=seq(0,2*pi,length=20)
> serie1=sin(ti)
> serie2=cos(ti)
> plot(ti,serie1,type='l',lty=1,xlab='temps',ylab='valeur')
> lines(ti,serie2,lty=2)
> legend("topleft",c("série 1","série 2"),lty=c(1,2))
```



Sauvegarder un graphique

```
> savePlot(filename="nom_fichier", type="jpeg")
```

Exemple de manipulation d'un tableau de données

- Charger les données iris

```
> data(iris)
```

- Classe des objets

```
> class(iris)
[1] "data.frame"
> class(iris$Species)
[1] "factor"
> class(iris$Sepal.Length)
[1] "numeric"
```

- Les variables présentes

```
> names(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

- Modalités de la variable qualitative "Species"

```
> levels(iris$Species)
[1] "setosa" "versicolor" "virginica"
```

Résumé de statistiques descriptives élémentaires

```
> summary(iris)

  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa     :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

Calcul de statistiques descriptives élémentaires

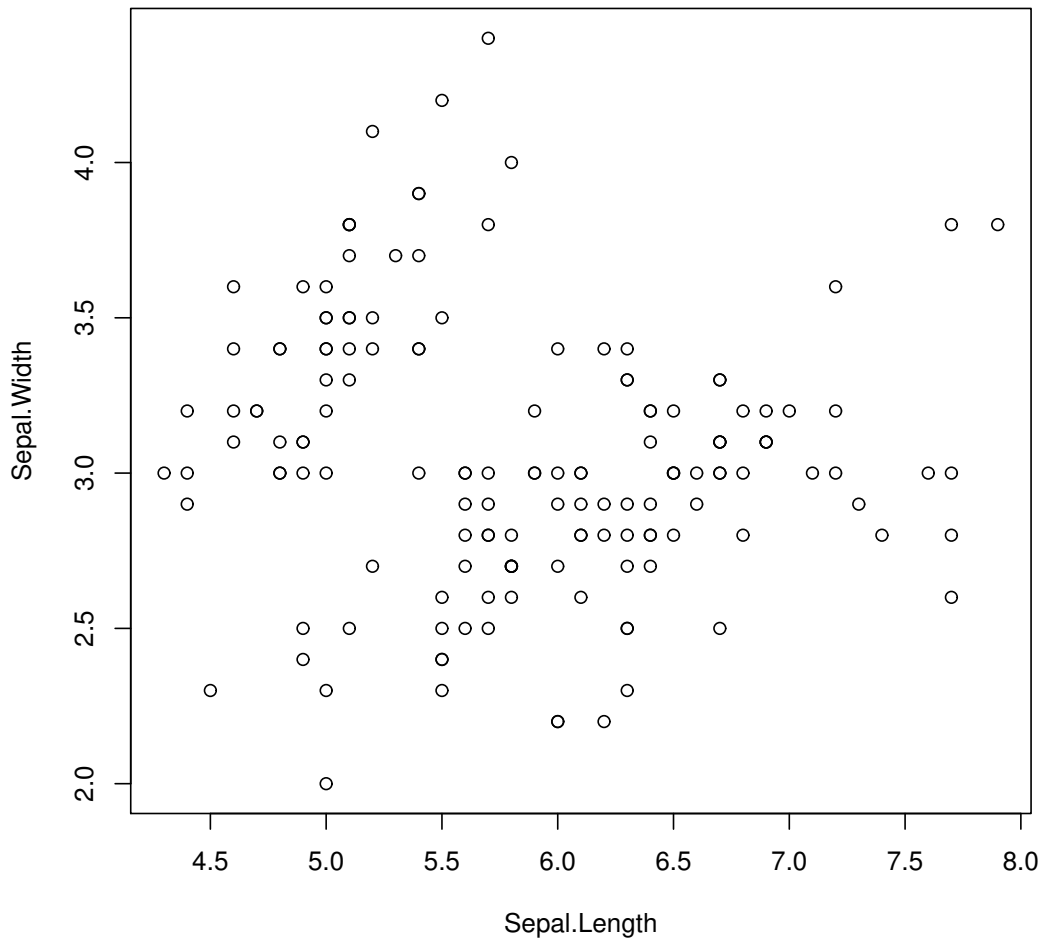
```
> mean(iris$Sepal.Length)  # moyenne
[1] 5.843333
> median(iris$Sepal.Length)  # mediane
[1] 5.8
> max(iris$Sepal.Length) # max
[1] 7.9
> min(iris$Sepal.Length) # min
[1] 4.3
> sd(iris$Sepal.Length)  # ecart-type
[1] 0.8280661
> cov(iris$Sepal.Length,iris$Petal.Length) # covariance
[1] 1.274315
> table(iris$Species)  # nombre d'occurrences des modalités
    setosa versicolor virginica
        50         50         50
```

Représentation graphique de variables quantitatives

```
> hist(iris$Sepal.length)
```

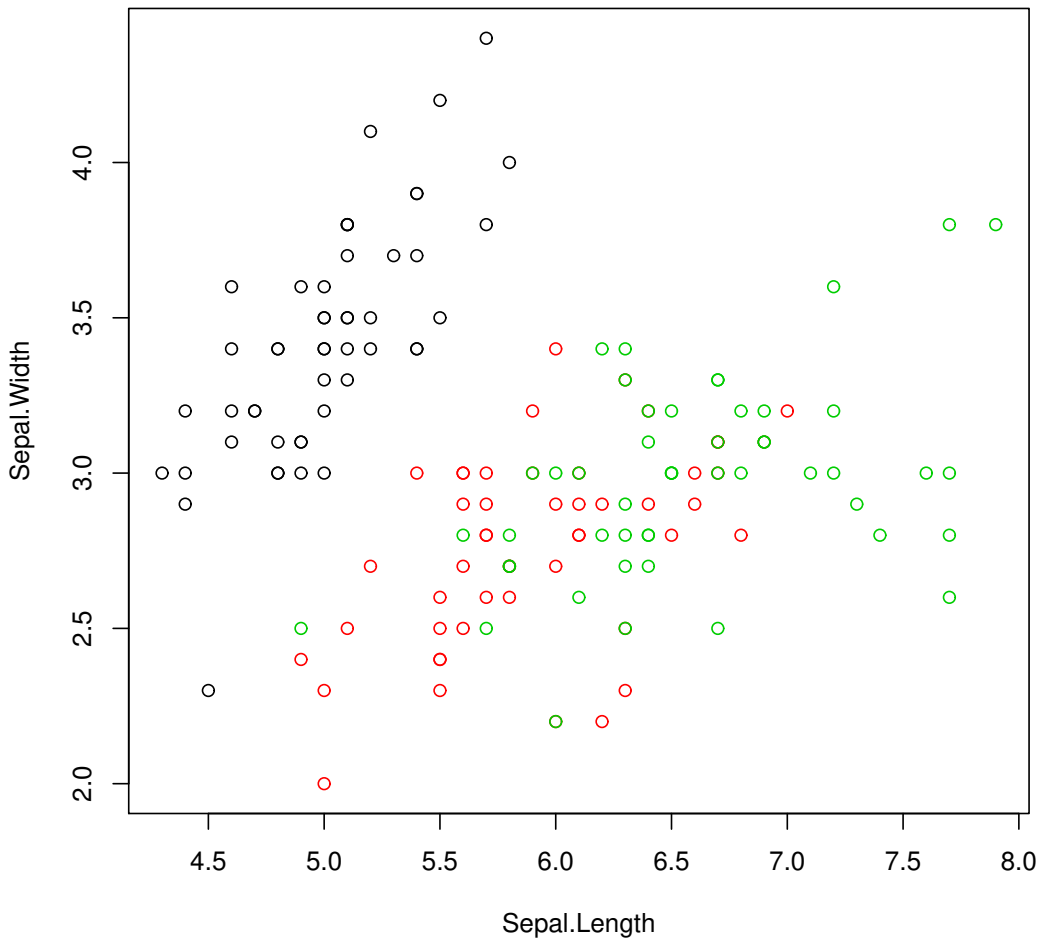
Représentation graphique de variables quantitatives

```
> plot(iris[c(1,2)])
```



Représentation graphique de variables quantitatives

```
> plot(iris[c(1,2)],col=c(1,2,3,4)[iris$Species])
```

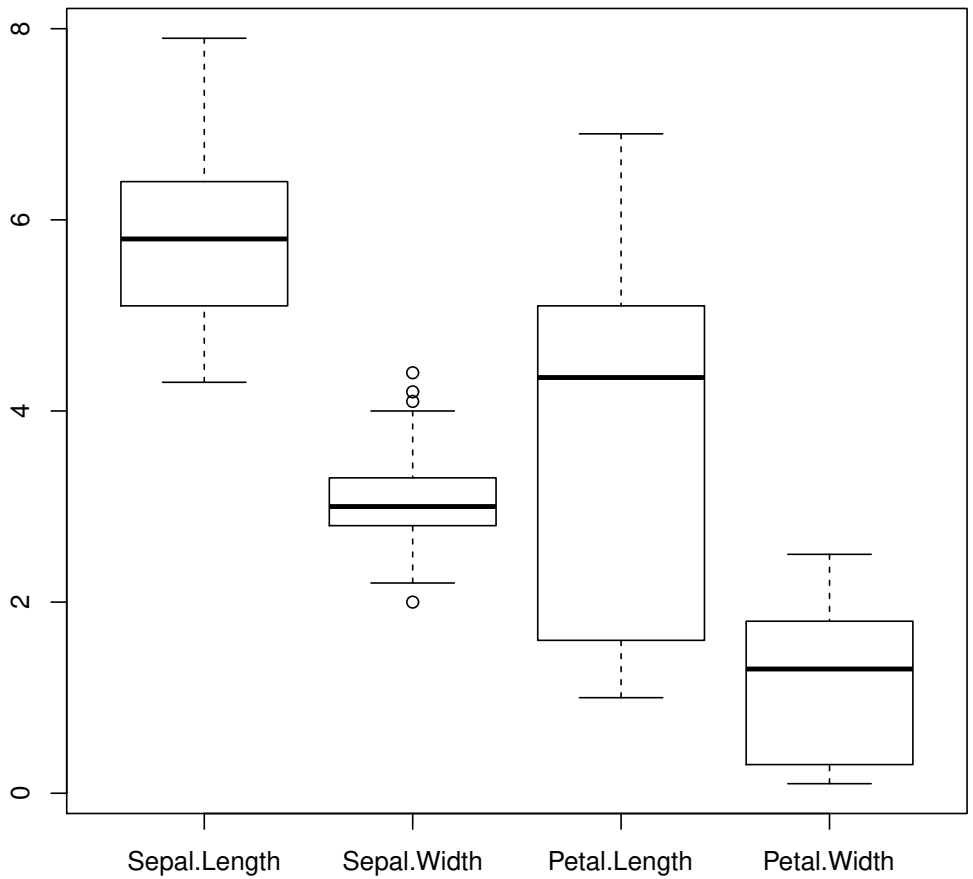


Représentation graphique de variables quantitatives

```
> hist(iris$Sepal.Length)
```

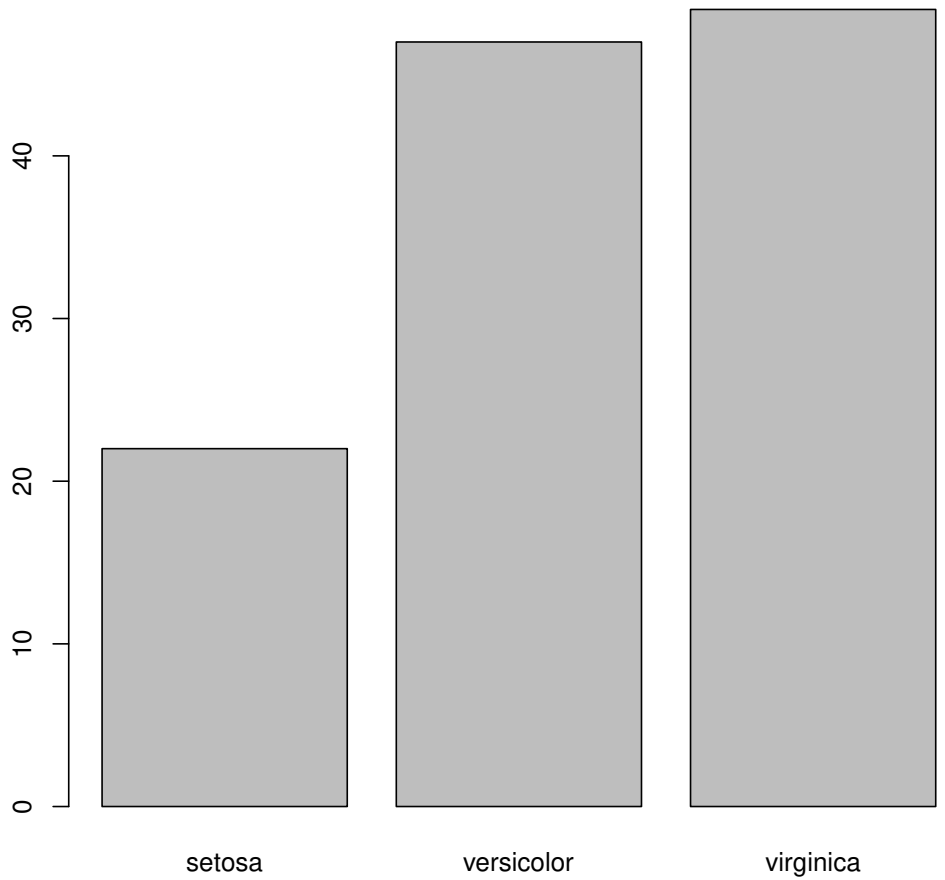
Représentation graphique de variables quantitatives : boxplot

```
> boxplot(iris[1:4])
```



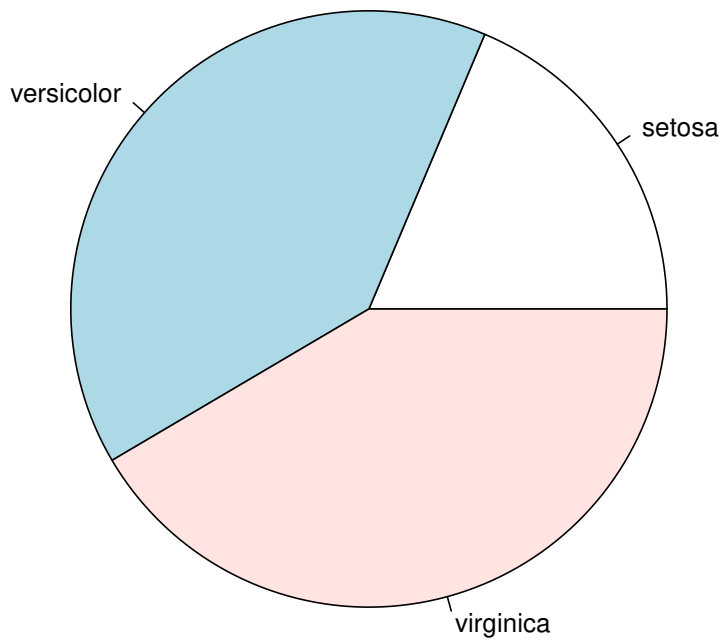
Représentation graphique de variables qualitative : diagramme en barre

```
> barplot(iris[1:4])
```



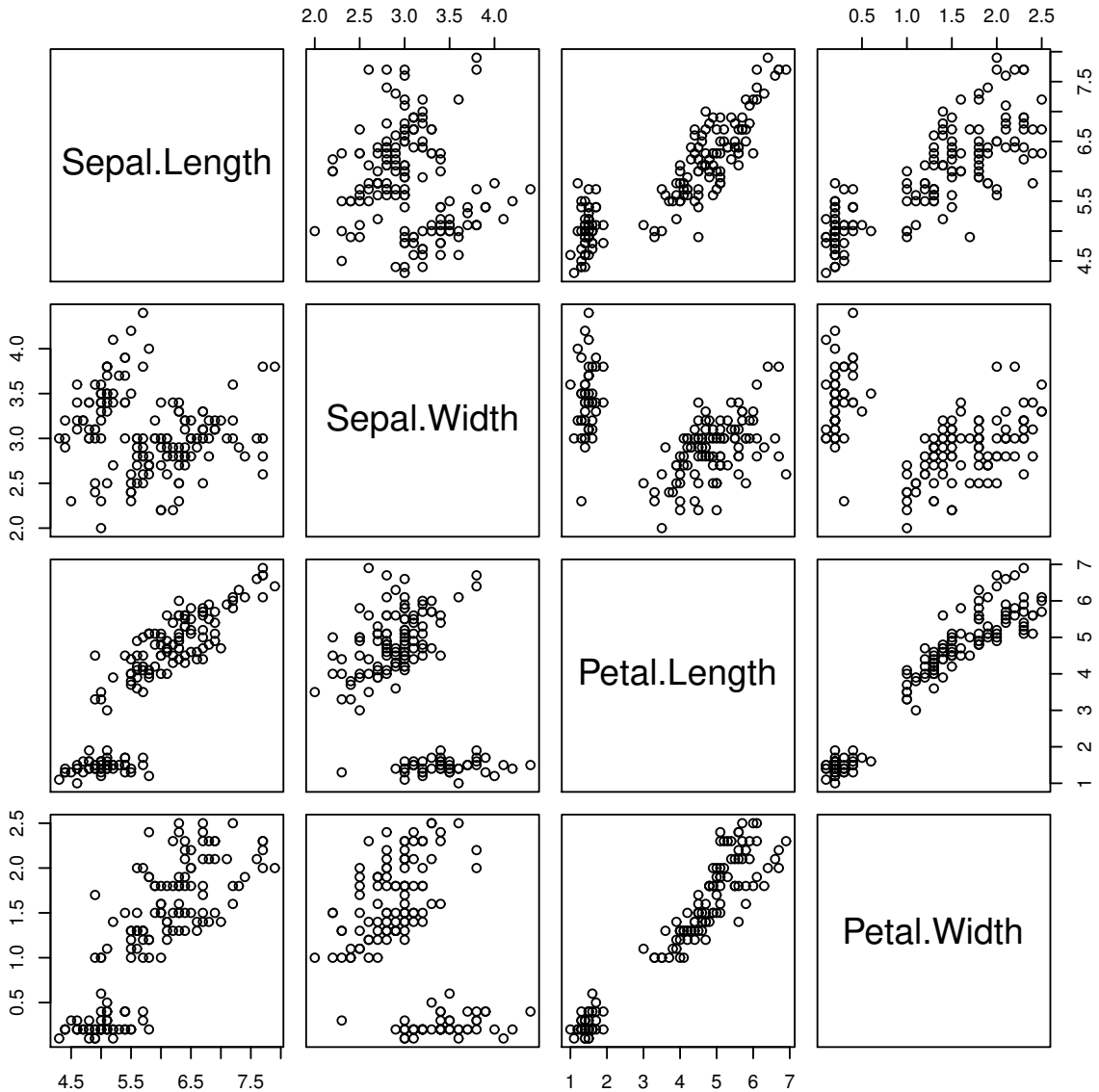
Représentation graphique de variables qualitative : camembert

```
> pie(iris[1:4])
```



Matrice de nuages de points

```
> pairs(iris[1:4])
```



Écrire des fonctions dans R

- Ecrire la fonction directement

```
> addition <- function(a,b)
+ {  add <- a+b
+   return(add)  }
```

- Écrire la fonction dans un fichier et la rendre connue du système

```
> source("monfichier.R")
```

- Utiliser la fonction

```
> addition(7,6)
[1] 13
```

- Inscrire les résultats des commandes dans un fichier

```
> sink("mes_resultats.txt")
> a <- c(1,2,3)
> addition(7,6)
> sink() # pour revenir à la console
```

Instructions de contrôle

- Boucle **For**

```
y <- 1:9
for (i in 1:length(y))
{
y[i] <- 0
}
```

- Instruction **If**

```
if (y[2] == 5)
{
y[i] <- 99
}
```

- Boucle **While**

```
i <- 1
while (i < length(y))
{
 y[i]  <- 999
 i <- i+1
 }
```

Commandes diverses

- Obtenir la liste de toutes les variables

```
> ls()
```

- Effacer une variable

```
> rm(nom_variable)
```

- Effacer toutes les variables

```
> rm ( list=ls() )
```

- Connaître la classe d'un objet

```
> class(nom_objet)
```

- Tracer une ligne horizontale ou verticale

```
> abline(h=2)
> abline(v=3)
```