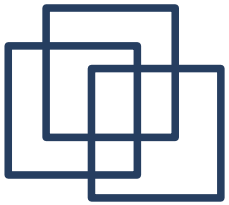
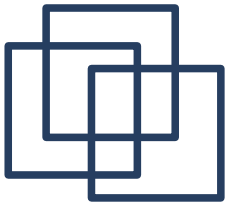


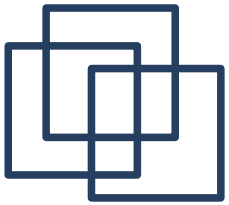
AIDE A LA DECISION

Le langage R



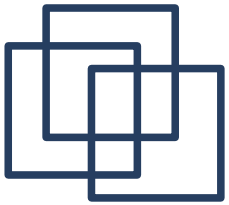
Ce que nous avons déjà vu

- Les objets simples (types : numeric, logical, factor)
- Les objets complexes : vector, matrix, data.frame, list
- Les opérateurs sur ces objets



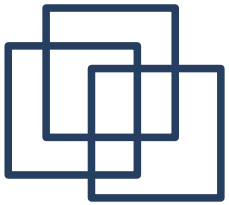
Ce que nous allons voir

- Compléments sur les data.frame, les fichiers de données et les graphiques
- Les éléments de programmation (boucles, conditionnelles, etc.)



Ce que nous allons voir

- Compléments sur les data.frame, les fichiers de données et les graphiques
- Les éléments de programmation (boucles, conditionnelles, etc.) → **A UTILISER AVEC MODERATION**



Compléments sur les data.frame

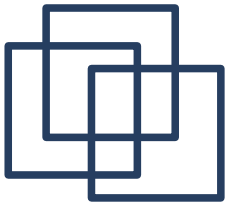
Rappel : un *data frame* est un tableau où

- 1 ligne = 1 individu (1 donnée)
- 1 colonne = 1 attribut (1 observation = 1 caractère)

Les attributs dont la valeur est manquante : NA

On importe facilement une telle structure à partir d'un fichier de données : texte, csv, xml, sql, oracle, xls, ods, ...

```
sommeil <- read.csv("sleep.csv" , dec="," , sep="")
```



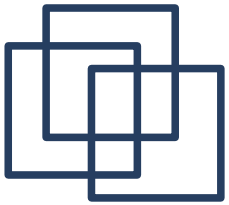
Compléments sur les data.frame

Fonctions de base sur les *data.frame* :

- nrow() : nombre de données
- ncol() : nombre d'attributs
- names() : noms des attributs
- summary() :statistiques descriptives élémentaires

3 syntaxes pour accéder aux données d'un *data.frame* Df:

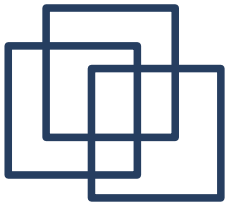
- Df [numéro(s) de ligne(s) , numéro(s) de colonne(s)]
- Df [numéro(s) de ligne(s) , nom(s) de colonne(s)]
- Df [numéro(s) de ligne(s),]$nom-de-colonne



Compléments sur les data.frame

Exercice :

- Obtenir la liste des « Species » de 3 manières
- Obtenir les noms des espèces 1, 4 et 5



Compléments sur les data.frame

Toutes les fonctions s'appliquent naturellement sur les sélections : min(), max(), mean(), median(), var(), sd()

exemple : min(sommeil$Brain.Weight)

En général, les NA doivent être éliminés : na.omit() ou option na.rm=T

exemple : min(na.omit(sommeil$Non.Dreaming))
min(sommeil$Non.Dreaming , na.rm=T)

Exercice : Quel est le mammifère qui dort le moins?

Compléments sur les data.frame

Toutes les fonctions s'appliquent naturellement sur les sélections : min(), max(), mean(), median(), var(), sd()

exemple : min(sommeil$Brain.Weight)

En général, les NA doivent être éliminés : na.omit() ou option na.rm=T

exemple : min(na.omit(sommeil$Non.Dreaming))
min(sommeil$Non.Dreaming , na.rm=T)

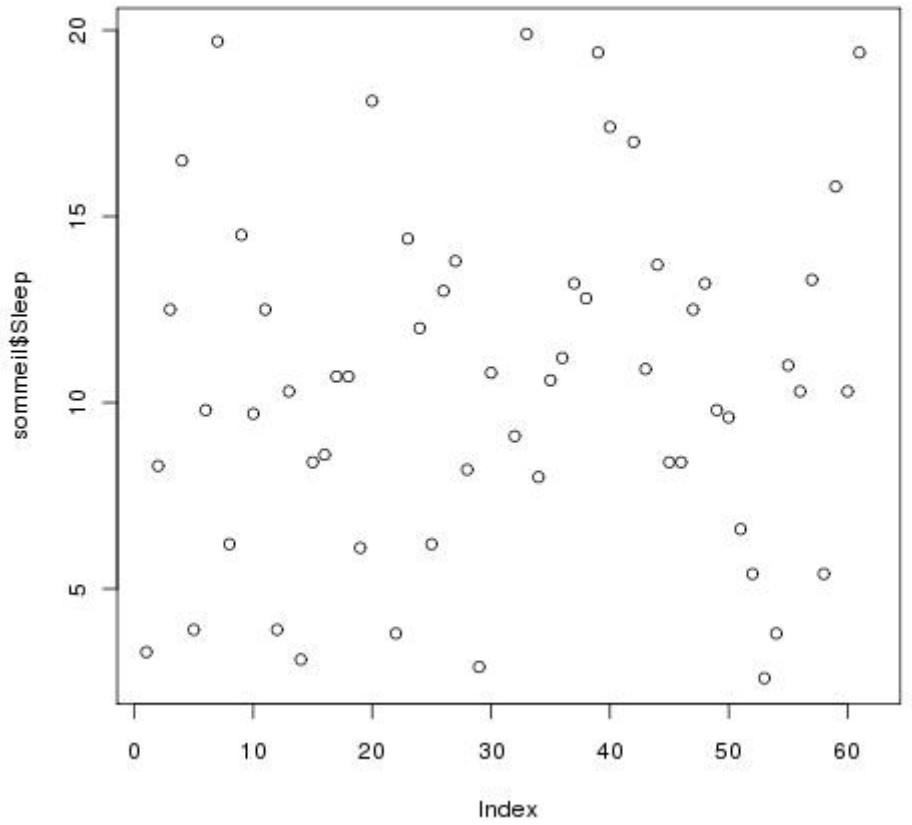
Exercice : Quel est le mammifère qui dort le moins?


© Jeff Moore

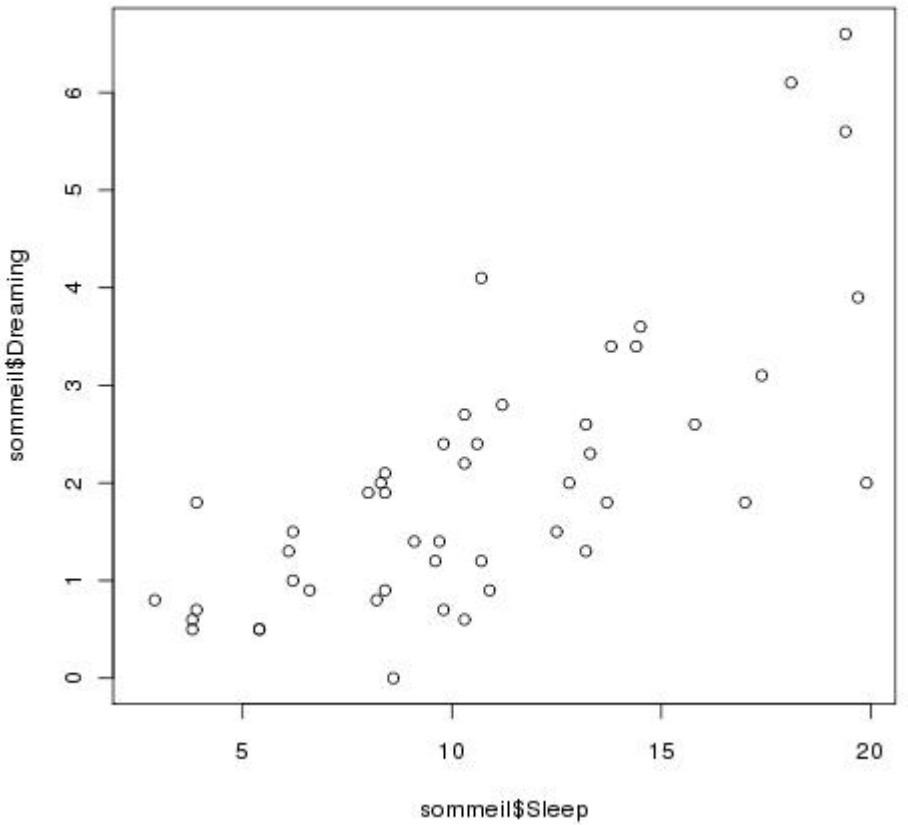
Graphiques simples

Graphique simple = graphe des valeurs d'attributs

plot(sommeil$Sleep)

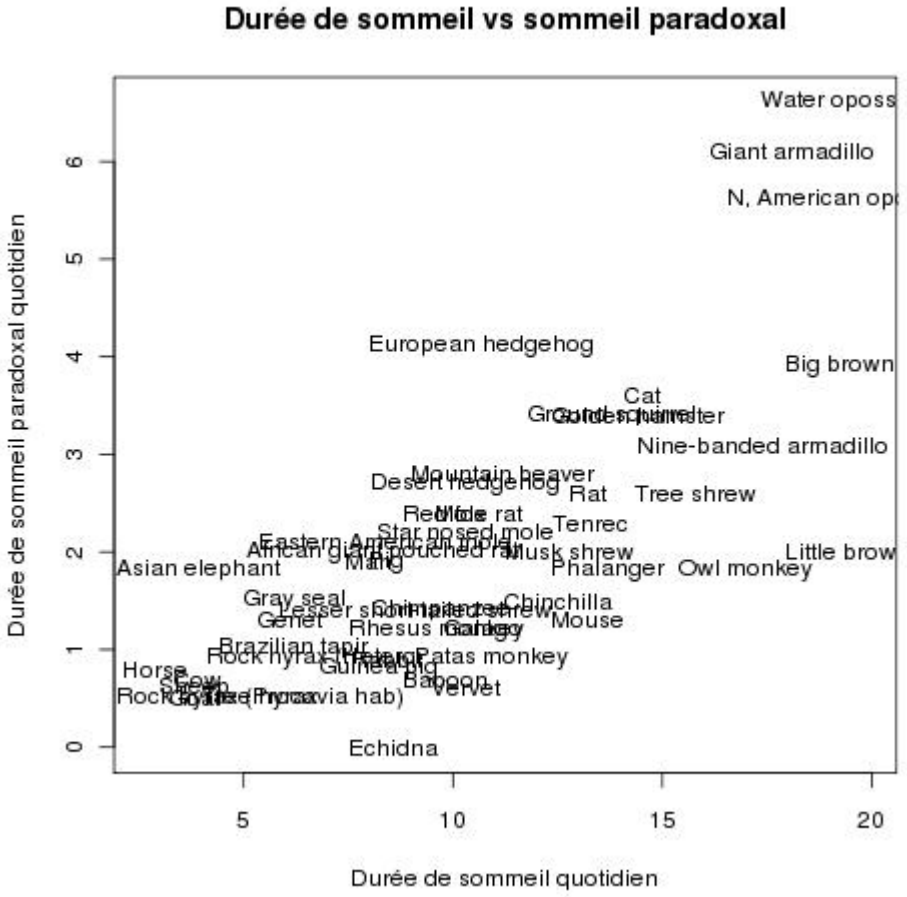


plot(sommeil$Sleep , sommeil$Dreaming)



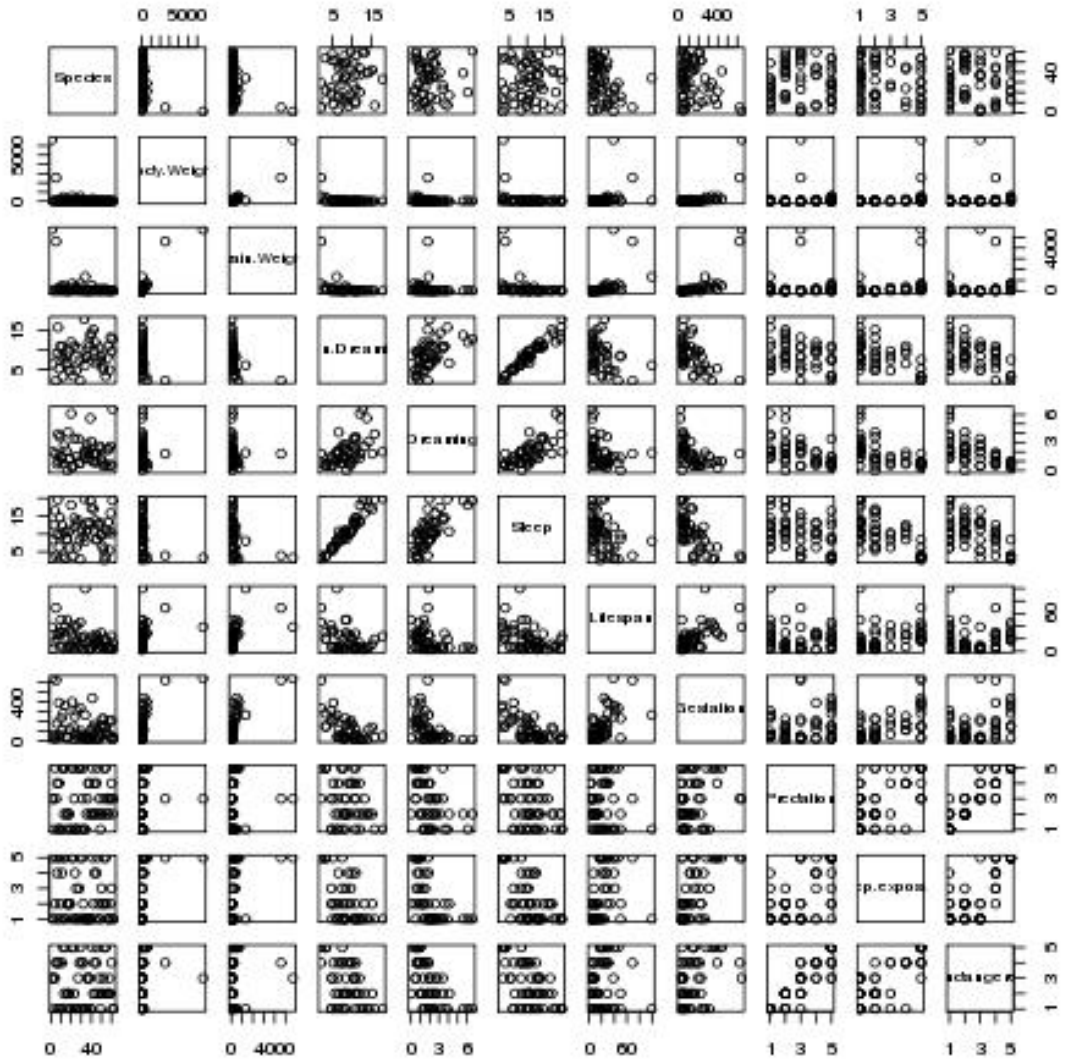
Graphiques simples avec titres

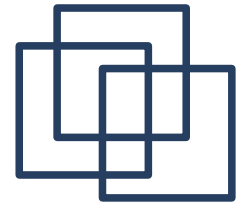
```
plot(sommeil$Sleep , sommeil$Dreaming , pch=' ' ,
      xlab="Durée de sommeil quotidien",
      ylab="Durée de sommeil paradoxal quotidien",
      main="Durée de sommeil vs sommeil paradoxal")
text(sommeil$Sleep , sommeil$Dreaming , sommeil$Species)
```



Graphiques simples avec titres

plot(sommeil)

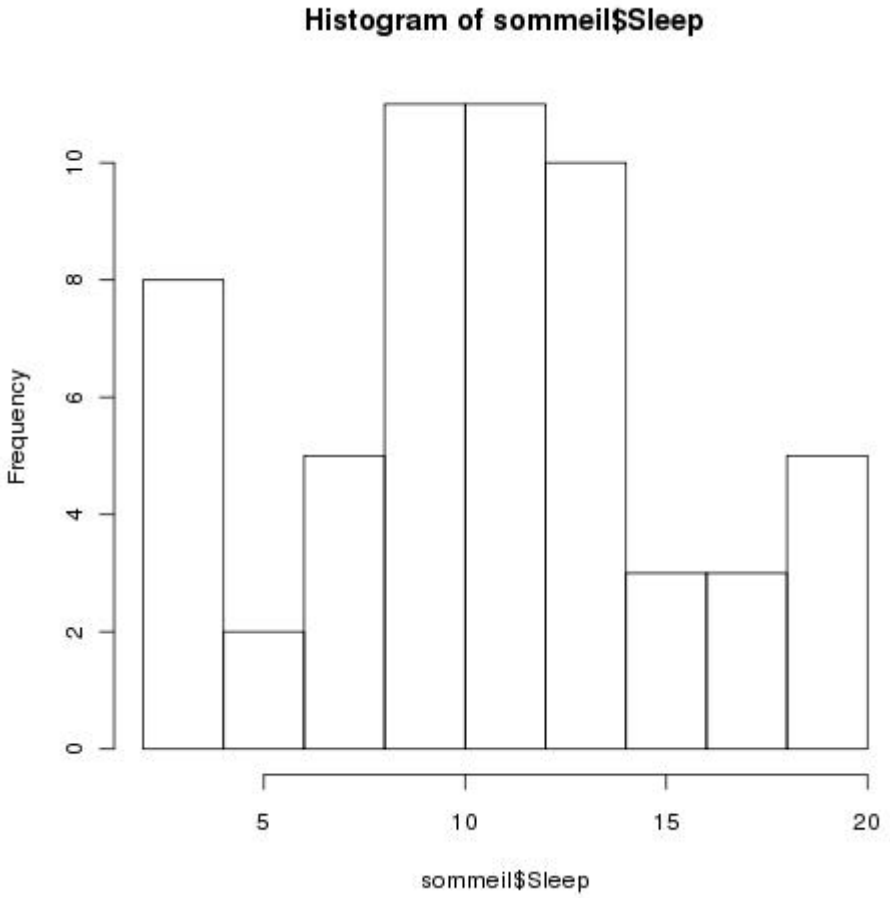




Histogrammes

Visualiser la distribution des valeurs d'un attribut

```
hist(sommeil$Sleep)
```

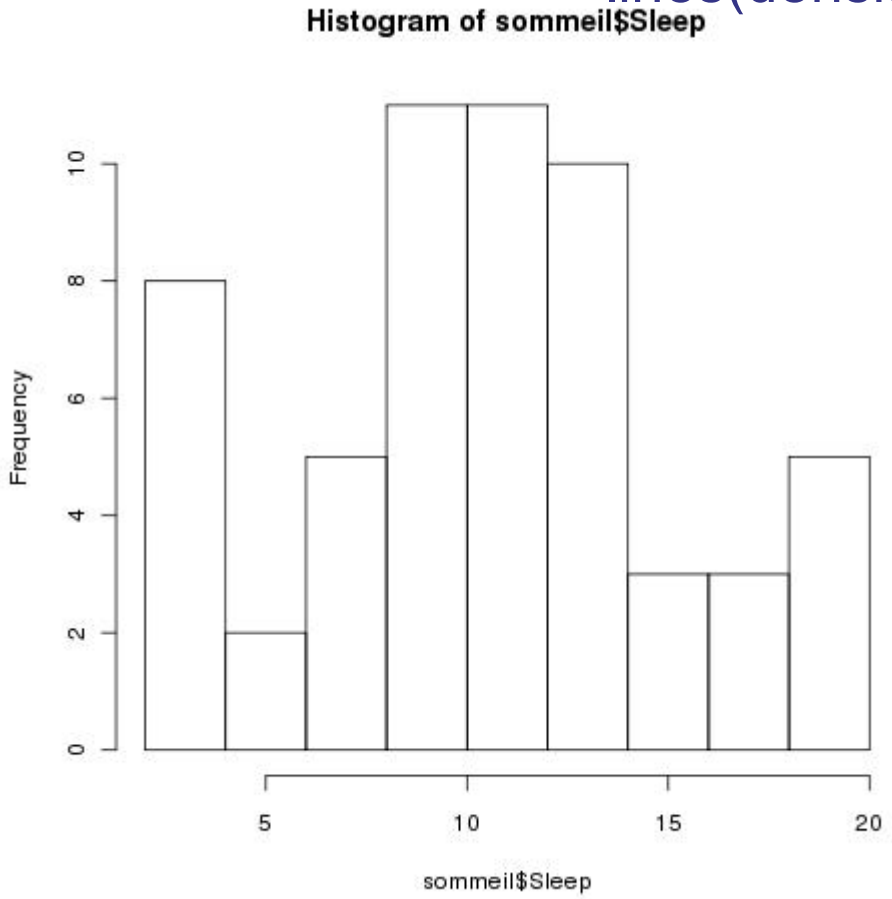


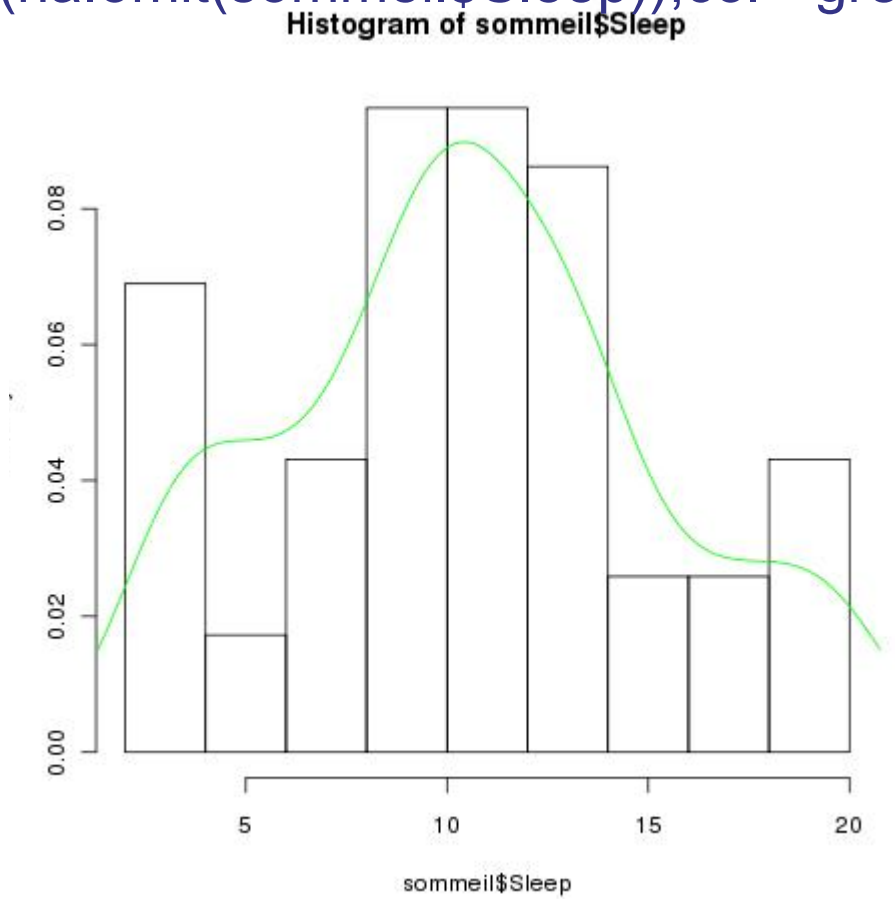
Histogrammes

Visualiser la distribution des valeurs d'un attribut

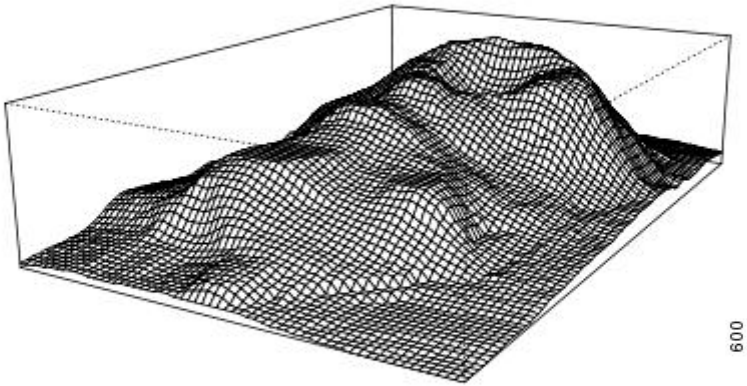
hist(sommeil$Sleep)

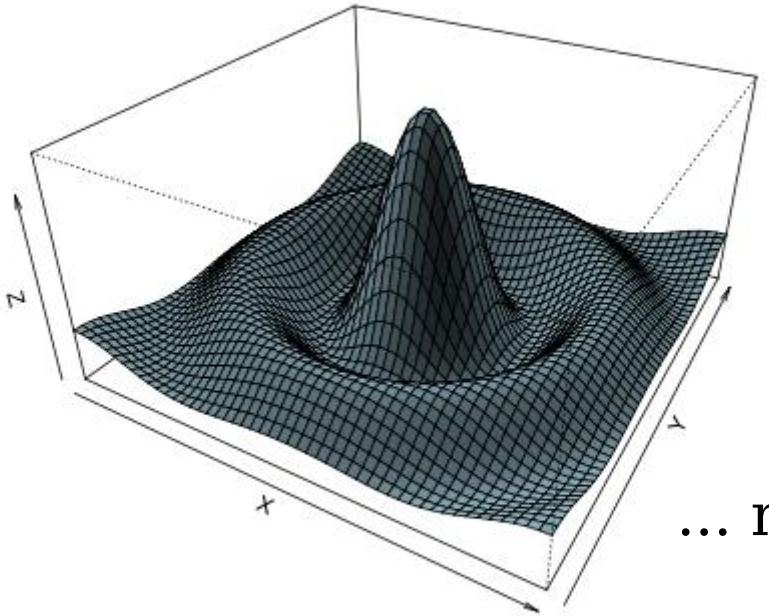
hist(sommeil$Sleep,freq=F)
lines(density (na.omit(sommeil$Sleep)),col="green")

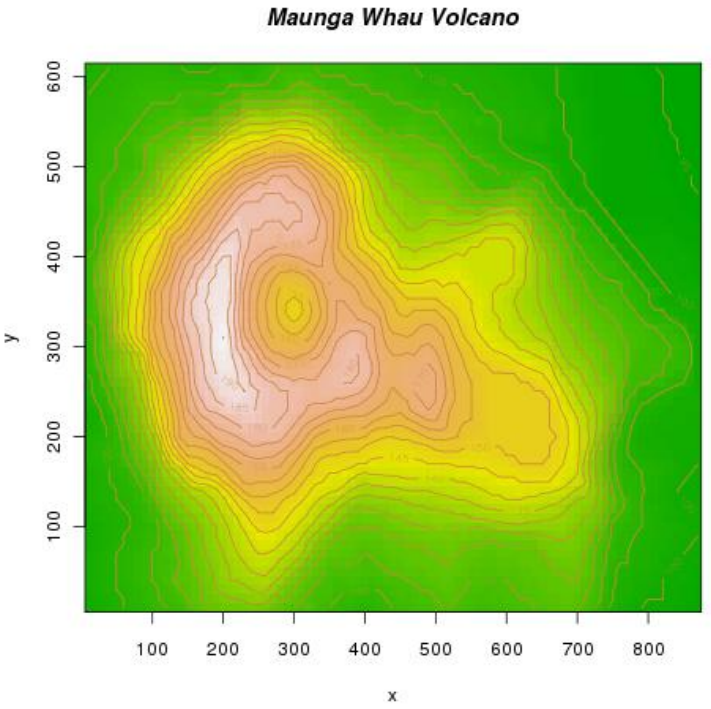




Pour aller plus loin...



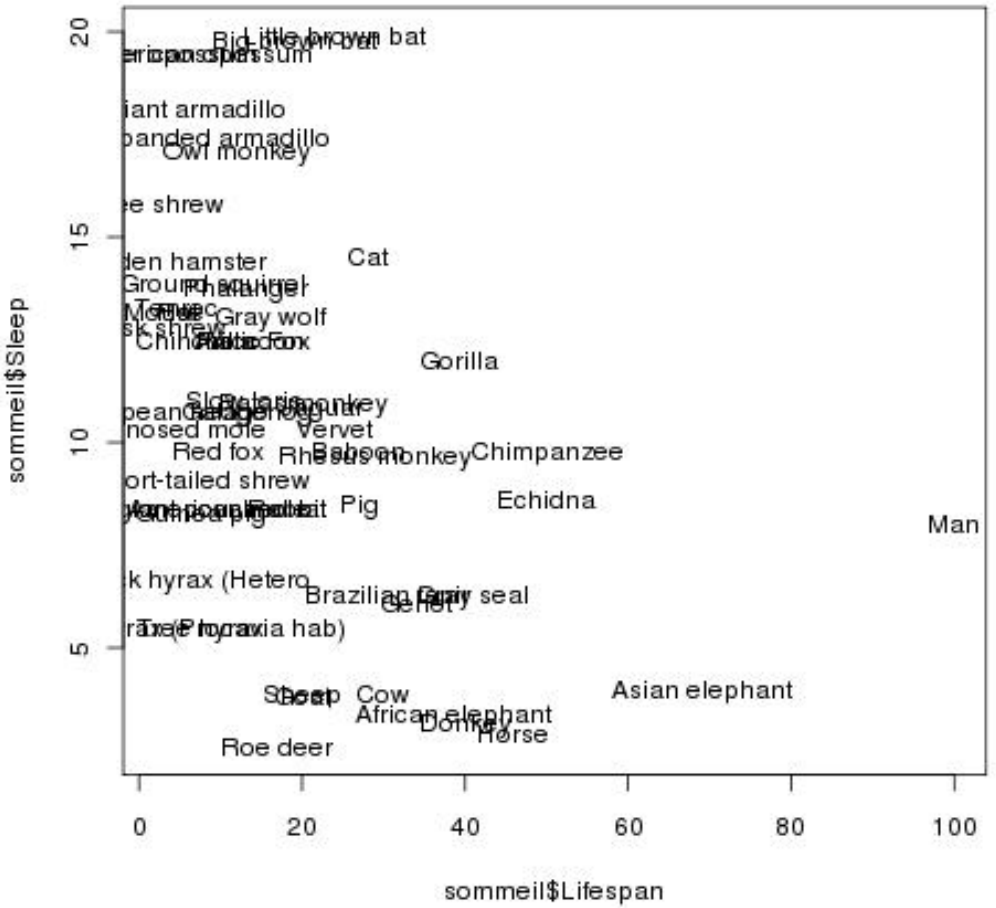




... mais R n'est pas un logiciel de dessin!!!

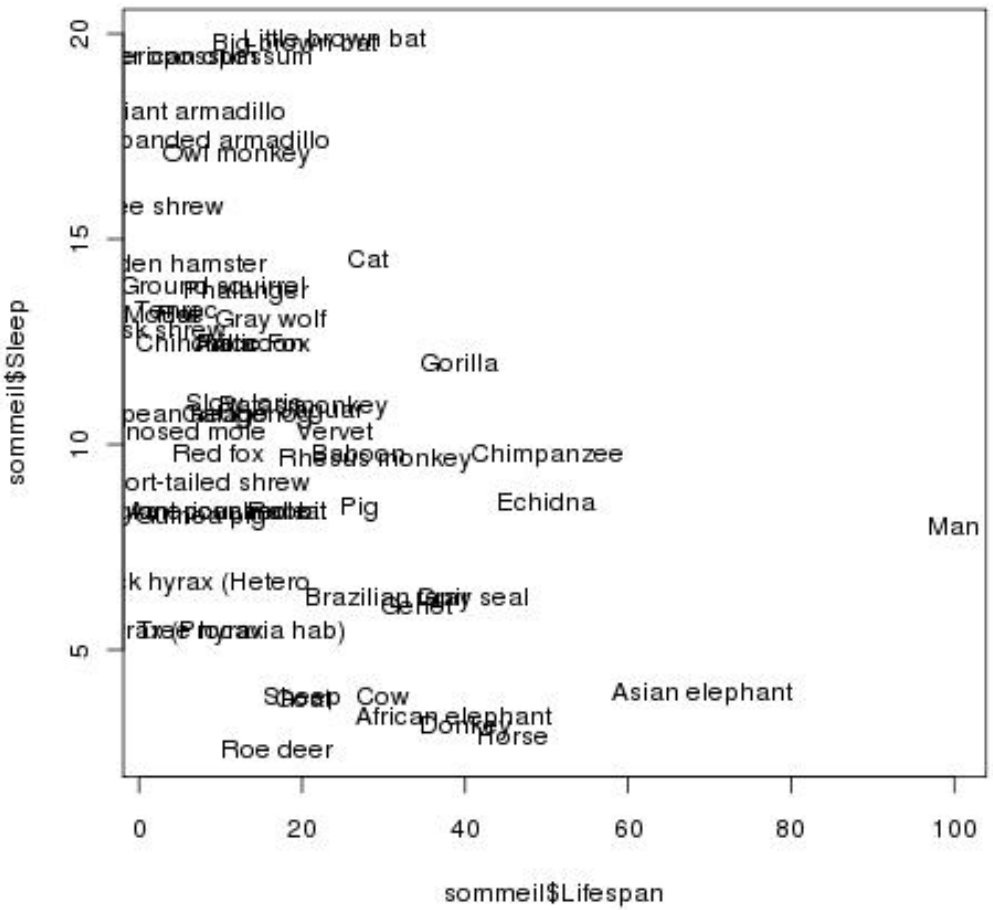
Petit pas vers l'AD

plot(sommeil$Lifespan,sommeil$Sleep)



Petit pas vers l'AD

```
plot(sommeil$Lifespan,sommeil$Sleep)
```



Restriction du data.frame :

```
som<-
na.omit(sommeil[,c("Species","Lifespan","Sleep")])
```

Construction d'une matrice de distances sur les deux attributs :

```
d<-dist(som[,c(2,3)])
```

Construction d'un dendrogramme :

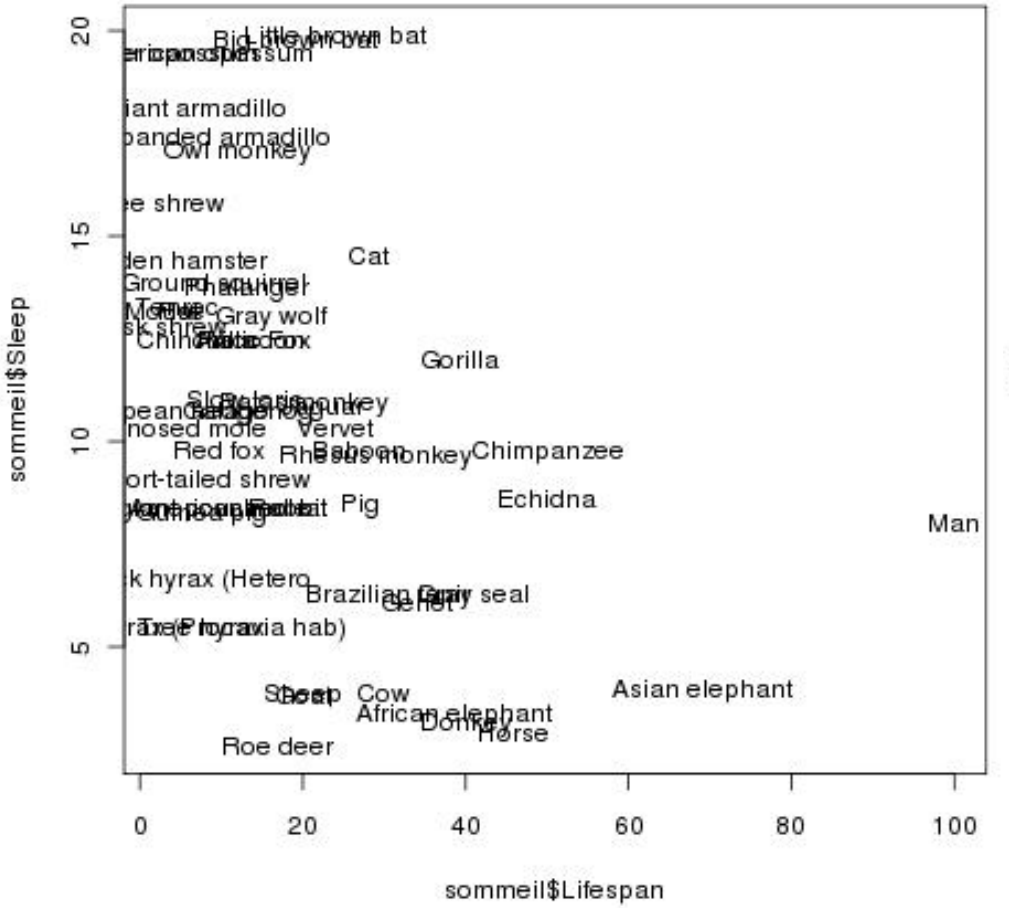
```
dendro<-hclust(d)
```

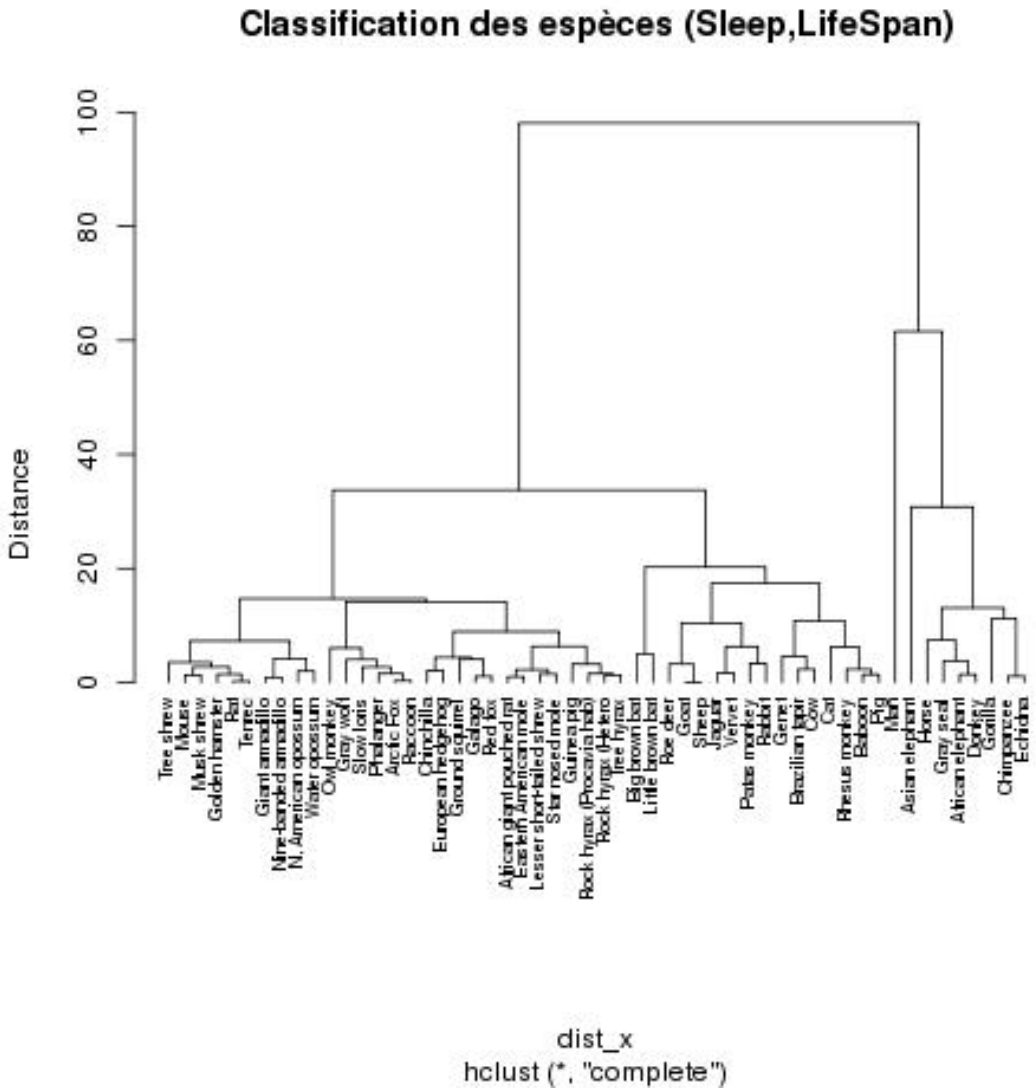
Affichage du dendrogramme :

```
plot(dendro,
    labels=som$Species,
    cex=.6,
    main="Classification des espèces...",
    ylab="Distance",
    hang=-1)
```

Petit pas vers l'AD

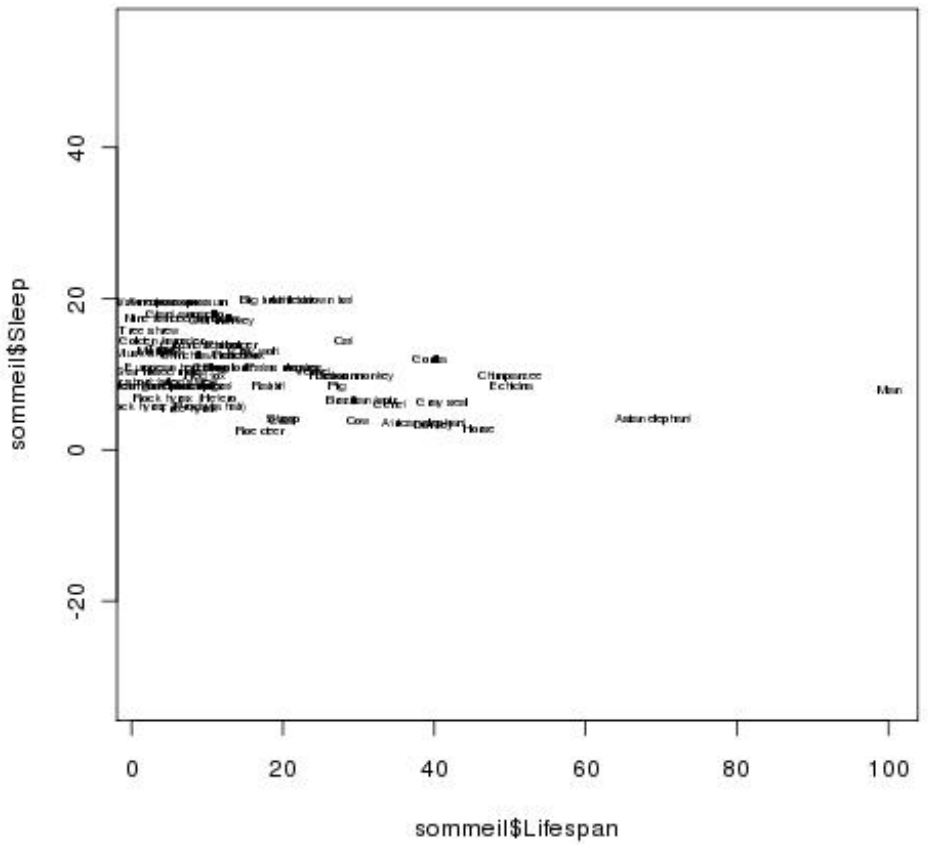
plot(sommeil$Lifespan,sommeil$Sleep)

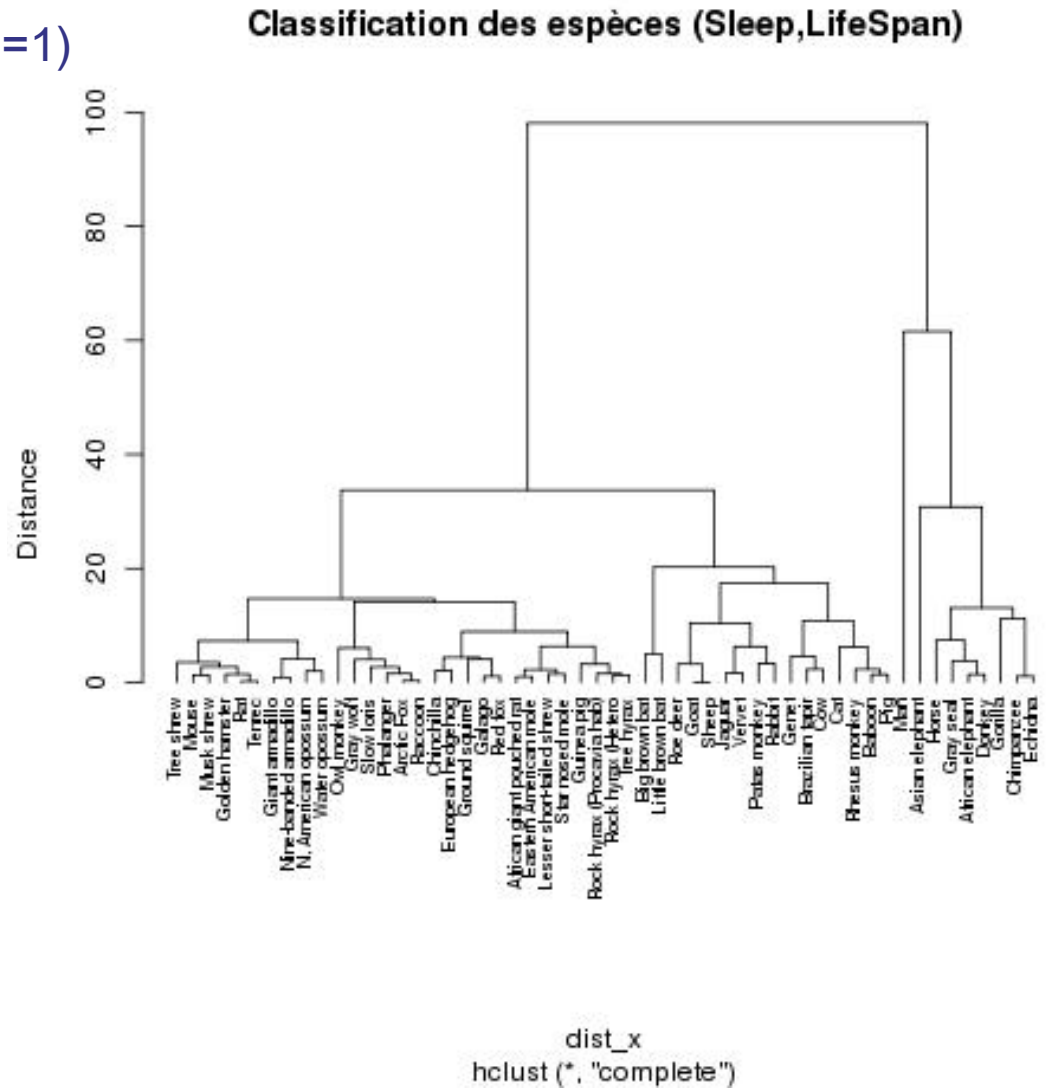


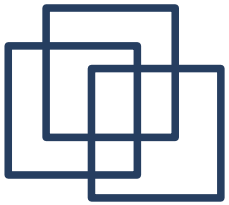


Petit pas vers l'AD

plot(sommeil$Lifespan,sommeil$Sleep,asp=1)

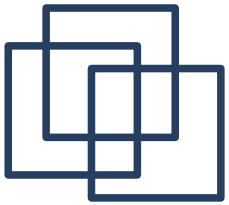






AIDE A LA DECISION

Le langage R : un langage de programmation



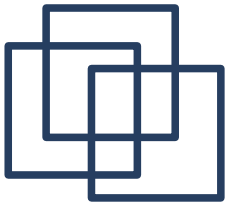
R : langage de programmation

R est (aussi) un langage de programmation très agréable à utiliser.

On peut y spécifier ses traitements avec des tests et des boucles ; on peut définir des fonctions.

On peut y définir des classes d'objets et des fonctions génériques.

Remarque : une bonne partie de R est écrite en R.



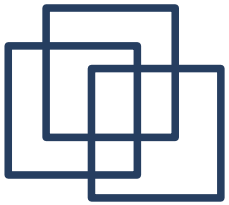
Structures de contôle

Syntaxe proche du Python : Itération (for, while), conditionnelle (if, switch)

- Boucle for : syntaxe

```
for(var in vector){
instructions
}
```

Instancie *var* avec chaque élément de *vector* successivement et exécute les instructions



Boucle for : exemples

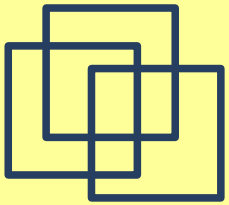
Calculer la somme des éléments d'un vecteur x

- À la C :

```
res<-0
for(i in 1:length(x)){
    res<-res+x[i]
}
```

- À la Perl :

```
res<-0
for(elt in x){
    res<-res+elt
}
```



Boucle for : exemples

Calculer la somme des éléments d'un vecteur x

- À la C :

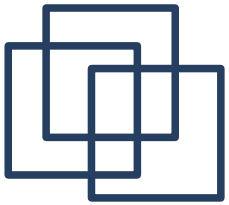
```
res<-0
for(i in 1:length(x)){
    res<-res+x[i]
}
```

- À la Perl :

```
res<-0
for(elt in x){
    res<-res+elt
}
```

Remarque : En R on préfèrera

```
sum(x)
```



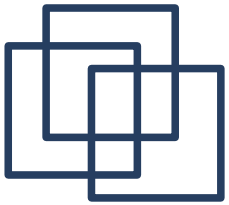
Conditionnelle

- L'instruction conditionnelle a la syntaxe générale suivante :

```
if( condition ){
    instructions
}
else {
    instructions
}
```

- Si la *condition* est évaluée à True le premier bloc d'instructions est exécuté, sinon le second bloc d'instructions est exécuté.

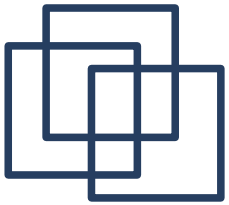
- Le *else* est optionnel.



Conditionnelle

Exercice : Soit un vecteur *x*, on souhaite construire un vecteur y de même taille et tel que :

- y[i]=1 si x[i]=b
- y[i]=0 si x[i]<>b



Conditionnelle

Exercice : Soit un vecteur *x*, on souhaite construire un vecteur y de même taille et tel que :

- y[i]=1 si x[i]=b
- y[i]=0 si x[i]<>b

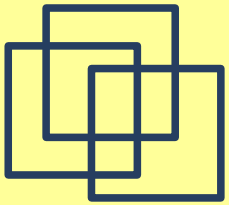
```
y<-numeric(length(x))
for(i in 1:length(x))
    if (x[i]==b) y[i]<-1
    else y[i]<-0
```

Conditionnelle

Exercice : Soit un vecteur *x*, on souhaite construire un vecteur y de même taille et tel que :

- y[i]=1 si x[i]=b
- y[i]=0 si x[i]<>b

```
y<-numeric(length(x))
for(i in 1:length(x))
    if (x[i]==b) y[i]<-1
    else y[i]<-0
```



Conditionnelle

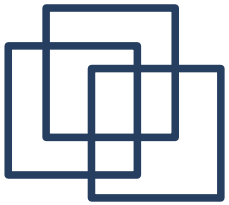
Exercice : Soit un vecteur *x*, on souhaite construire un vecteur y de même taille et tel que :

- y[i]=1 si x[i]=b
- y[i]=0 si x[i]<>b

```
y<-numeric(length(x))
for(i in 1:length(x))
    if (x[i]==b) y[i]<-1
    else y[i]<-0
```

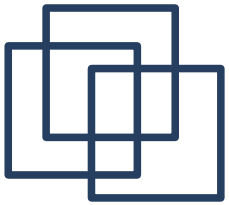
Remarque : En R on peut faire beaucoup plus directement...

```
y<-as.numeric(x==b)
```



AIDE A LA DECISION

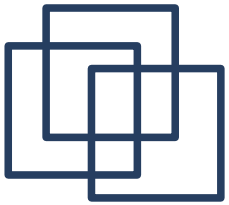
Le langage R : Écrire un programme en R



Fichier .R

Un programme en R sera écrit dans un fichier avec l'extension .R : utile pour exécuter plusieurs fois un même enchaînement de tâches.

Exécution via la commande : source("nomfic.R")



Les fonctions

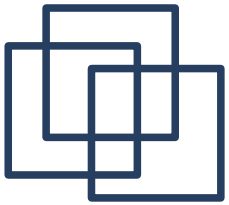
L'essentiel du travail en R se fait à l'aide de fonctions. L'utilisateur peut écrire ses propres fonctions...

- Syntaxe des fonctions :

```
nom_fonction <- function(liste_arguments){
    opérations
}
```

- Exercice :

écrire une fonction qui ajoute 1 à chaque élément d'un vecteur



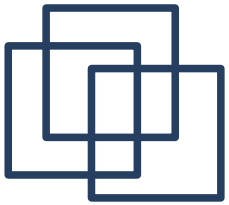
Pause « time »

Comparaison de l'efficacité de deux solutions

```
plus1_vec<-function(x){
    y<-as.numeric(length(x)) ;
    for(i in 1:length(x)) y[i]=x[i]+1 ;
    return(y) ;}

y<-plus1_vec(x);
```

Et encore mieux :

```
plus1<-function(x){return(x+1);}
y<-sapply(x,plus1);
```



Pause « time »

Comparaison de l'efficacité de deux solutions

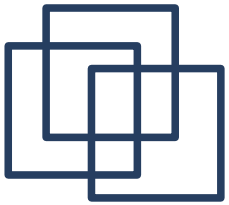
```
plus1_vec<-function(x){
    y<-as.numeric(length(x)) ;
    for(i in 1:length(x)) y[i]=x[i]+1 ;
    return(y) ;}

y<-plus1_vec(x);
```

34.40s sur un vecteur de taille 100,000

Et encore mieux :

```
plus1<-function(x){return(x+1);}
y<-sapply(x,plus1);
```

0.40s sur un vecteur de taille 100,000



Pause « time »

Comparaison de l'efficacité de deux solutions

```
plus1_vec<-function(x){
    y<-as.numeric(length(x)) ;
    for(i in 1:length(x)) y[i]=x[i]+1;
    return(y) ;}

y<-plus1_vec(x);
```

0.31s sur un vecteur de taille 10,000

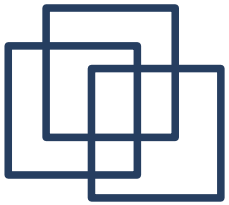
y<-plus1(x)

0.001s sur un vecteur de taille 100,000!!!

Et encore mieux :

```
plus1<-function(x){return(x+1);}
y<-sapply(x,plus1);
```

0.04s sur un vecteur de taille 10,000



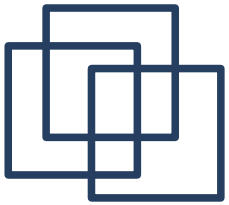
Les fonctions

Arguments :

- On spécifie les arguments lors d'un appel : par leur position ou par leur nom
- Possibilité d'utiliser les valeurs par défaut des arguments (arguments optionnels)

Portée des variables :

- Si un nom de variable est inconnu dans une fonction, R recherche dans l'environnement « immédiatement » supérieur
- La déclaration d'une nouvelle variable de même nom « cache » la variable précédente



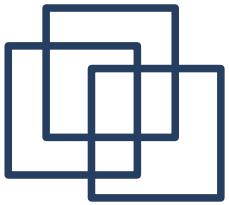
Les fonctions : exemples

Soit la définition de fonction suivante :

```
foo <- function(arg1, arg2=5, arg3=FALSE)
```

Parmi les appels suivants, lesquels sont valides?

- foo(x,y,z)
- foo(x)
- foo(arg3=z , arg1=x , arg2=y)
- foo(arg2=y , arg1=x)
- foo(arg1=x)
- foo(arg1=y)
- foo(arg2=y)

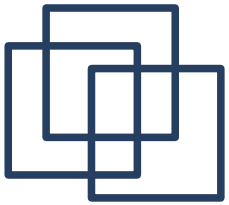


Les fonctions : exemples

Quels sont les affichages produits par ces programmes R ?

```
foo <- function() print(x)
x<-1
foo()
```

```
x<-1
foo <- function() {x<-2 ; print(x)}
foo()


x
```

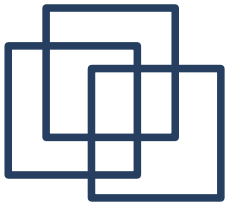


Les fonctions : exemples

Quels sont les affichages produits par ses programmes R ?

```
foo <- function() print(x)
x<-1
foo()
[1] 1
```

```
x<-1
foo <- function() {x<-2 ; print(x)}
foo()
[1] 2

x
[1] 1
```



Les fonctions apply

Ces fonctions permettent d'appliquer une même fonction à l'ensemble des éléments d'une liste, d'un vecteur, d'une matrice, etc.

fonction	Structure concernée	Sortie
apply()	tableau (array)	vecteur, tableau ou liste
sapply()	vecteur ou liste	vecteur ou matrice
lapply()	vecteur ou liste	liste
mapply()	listes ou vecteurs multiples	liste, vecteur ou matrice