ISV51: Programmation sous R

Structure de données

L3 GBI – Université d"Evry

semestre d'automne 2015

`http://julien.cremeriefamily.info/teachings_L3BI_ISV51.html`

Laboratoire de Mathématiques et Modélisation d'Évry

LaMME

université evry val-d'essonne

Plan

Variables et types élémentaires

Vecteurs

Facteurs

Matrices (et tableaux)

Listes

Tableau de données

Plan

Variables et types élémentaires
- Définition
- Manipulation

Vecteurs

Facteurs

Matrices (et tableaux)

Listes

Tableau de données

Plan

Variables et types élémentaires
- Définition
- Manipulation

Vecteurs

Facteurs

Matrices (et tableaux)

Listes

Tableau de données

Déclaration d'une variable

En R, affecter une variable revient à la déclarer.

Définition (affectation)

Opération consistant à attribuer une valeur à une variable.

Plusieurs choix possibles

- l'opérateur usuel est '<-' (signe inférieur suivi de moins)

```
jo <- "l'indien"
jo
```

```
## [1] "l'indien"
```

- l'opérateur '=' peut être utilisé

```
nb.max.d.annees.pour.faire.une.these = 3
nb.max.d.annees.pour.faire.une.these
```

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

- la commande `assign` (d'où l'anglicisme *assignation*)

```
assign("x", -pi)
x
```

```
## [1] -3.141593
```

Les principaux type et modes

Types et modes

Pour une variable élémentaire (atomique), on a principalement

type	`typeof`	**mode**	`mode`
flottant	`double`	numérique	`numeric`
entier	`integer`	numérique	`numeric`
complexe	`complex`	complex	`complex`
booléen	`logical`	logique	`logical`
caractère	`caracter`	caractère	`caracter`

Types et modes II

D'autres types sont possible pour des objets plus complexes

type	`typeof`	**mode**	`mode`
liste	`list`	liste	`list`
NULL	`NULL`	NULL	`NULL`
closure	`closure`	fonction	`function`
raw	`raw`	raw	`raw`

Les principaux type et modes

Types et modes

Pour une variable élémentaire (atomique), on a principalement

type	`typeof`	**mode**	`mode`
flottant	`double`	numérique	`numeric`
entier	`integer`	numérique	`numeric`
complexe	`complex`	complex	`complex`
booléen	`logical`	logique	`logical`
caractère	`caracter`	caractère	`caracter`

Types et modes II

D'autres types sont possible pour des objets plus complexes

type	`typeof`	**mode**	`mode`
liste	`list`	liste	`list`
NULL	`NULL`	NULL	`NULL`
closure	`closure`	fonction	`function`
raw	`raw`	raw	`raw`

Valeurs spéciales, réservées par R

- `TRUE`/`FALSE`, indicateurs logiques
- `NA`, valeurs manquantes, de type `logical`
- `NaN`, résultat numérique aberrant, de type `double`
- `Inf` et `-Inf`, plus et moins ∞, de type `double`
- `NULL`, l'objet nul, de type `NULL`

```
c(4,2,NA,5)
```

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

```
0/0
```

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

```
1/0
```

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

```
names(1)
```

```
## NULL
```

Plan

Variables et types élémentaires

- Définition
- Manipulation

Vecteurs

Facteurs

Matrices (et tableaux)

Listes

Tableau de données

Variable atomique numérique

Opérateurs arithmétiques élémentaires '+','-','/','*'

```
x <- pi
2*x

## [1] 6.283185

mode(x)

## [1] "numeric"

typeof(x)

## [1] "double"
```

ainsi que ^,%%,%/%,abs,log,exp,log10,sqrt,cos,tan,sin

Variable atomique logique

Opérateurs logiques élémentaires `'&','|','!','xor'`

```
x <- TRUE; y <- FALSE
x | y

## [1] TRUE

mode(x)

## [1] "logical"

typeof(x)

## [1] "logical"
```

Chaîne de caractères

Opérations élémentaires : `'paste','substr','sub'` et bien d'autres

```
x <- "allo"; y <- "non mais"
paste(y,x)

## [1] "non mais allo"

substr(y,5,8)

## [1] "mais"
```

Tester le type d'une variable

Les fonctions de format `is.xx` permet de tester le type d'une variable

```
is.numeric(pi)

## [1] TRUE

is.logical("chaîne")

## [1] FALSE

is.character("")

## [1] TRUE

is.integer(1) ## étonnant ?

## [1] FALSE

is.null(NULL)

## [1] TRUE
```

Convertir une variable

La fonction générique as et ses dérivées le permet

```
is.integer(as.integer(1))
```

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

```
as(TRUE, "numeric")
```

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

```
as.character(pi)
```

```
## [1] "3.14159265358979"
```

On peut aussi forcer le type !

```
x <- 2*pi; mode(x) <- "character"; x
```

```
## [1] "6.28318530717958"
```

```
x <- 2*pi; mode(x) <- "logical"; x
```

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

Plan

Variables et types élémentaires

Vecteurs
- Définition
- Opérations élémentaires
- Génération de vecteurs
- Manipulation
- Représentation graphique minimal

Facteurs

Matrices (et tableaux)

Listes

Tableau de données

Plan

Variables et types élémentaires

Vecteurs
- Définition
- Opérations élémentaires
- Génération de vecteurs
- Manipulation
- Représentation graphique minimal

Facteurs

Matrices (et tableaux)

Listes

Tableau de données

Vecteurs : définition

Propriétés

- objet le plus élémentaire sous `R`,
- collection d'entités de même nature,
- mode défini par la nature des entités qui le composent.

Création par initialisation I

Avec les fonctions issus des types :

```
integer(5)

## [1] 0 0 0 0 0

double(5)

## [1] 0 0 0 0 0

character(5)

## [1] "" "" "" "" ""

logical(5)

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

Création par initialisation II

Avec la commande `vector`, trop souvent oubliée

```
v <- vector(mode="numeric", 3); v

## [1] 0 0 0

v <- vector(mode="logical", 4); v

## [1] FALSE FALSE FALSE FALSE

v <- vector(mode="character", 5); v

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

Création par concaténation I

En « combinant » des éléments de même type à l'aide de la fonction `c()`

1. Numérique

```
x1 <- c(-1,23,98.7)
mode(x1)

## [1] "numeric"
```

2. Caractère

```
y1 <- c("Pomme","Flore","Alexandre")
mode(y1)

## [1] "character"
```

3. Logique

```
z1 <- c(FALSE,TRUE,FALSE,TRUE,TRUE)
z2 <- c(T,F,F)
mode(z2)

## [1] "logical"
```

Création par concaténation II

Attention : un seul mode possible...

```
c(TRUE,"bonjour", 2)
```

```
## [1] "TRUE"    "bonjour" "2"
```

mais

```
"bonjour"
```

```
## [1] "bonjour"
```

Plan

Variables et types élémentaires

Vecteurs
- Définition
- Opérations élémentaires
- Génération de vecteurs
- Manipulation
- Représentation graphique minimal

Facteurs

Matrices (et tableaux)

Listes

Tableau de données

Opérations arithmétiques I

s'effectuent terme-à-terme

Soient `x,y` de mode `numeric` tels que

```
x<-c(1,2,-3,-4)
y<-c(-5,-6,9,0)
```

- `'+'` addition des éléments de deux vecteurs

```
x+y
```

```
## [1] -4 -4  6 -4
```

- `'-'` soustraction des éléments de deux vecteurs

```
x-y
```

```
## [1]   6   8 -12  -4
```

- `'*'` multiplication des éléments de deux vecteurs

```
x*y
```

```
## [1]  -5 -12 -27   0
```

Opérations arithmétiques II

s'effectuent terme-à-terme

'/' division des éléments de deux vecteurs

```
x/y
```

```
## [1] -0.2000000 -0.3333333 -0.3333333       -Inf
```

'%%' division entière

```
abs(x) %% abs(y)
```

```
## [1]   1   2   3 NaN
```

'%/%' reste de la division entière

```
abs(y) %/% abs(x)
```

```
## [1] 5 3 3 0
```

Le « recyclage » des éléments du vecteur

Lors d'une opération entre vecteurs, les vecteurs trop courts sont ajustés pour atteindre la taille du plus grand vecteur en recyclant les données.

Exemple

```
x <- c(10,100,1000)
y <- c(1,2)
x+10
```

```
## [1]   20  110 1010
```

```
2*x + y - 1
```

```
## Warning in 2 * x + y: la taille d'un objet plus long n'est pas multiple de
la taille d'un objet plus court
```

```
## [1]   20  201 2000
```

$\rightsquigarrow$ souvent pratique mais attention aux effets de bords !

Opérateurs mathématiques I

Fonctions numériques élémentaires

`floor,ceiling,round`.

Opérateurs mathématiques II

```
x<-c(1,2,-3,-4); y<-c(-5,-6,9,0)
x/y
```

```
## [1] -0.2000000 -0.3333333 -0.3333333       -Inf
```

```
floor(x/y)
```

```
## [1]   -1   -1   -1 -Inf
```

```
ceiling(x/y)
```

```
## [1]    0    0    0 -Inf
```

```
round(x/y,3)
```

```
## [1] -0.200 -0.333 -0.333   -Inf
```

Opérateurs mathématiques III

Fonctions arithmétiques élémentaires

`^`,`%%`,`%/%`,`abs`,`log`,`exp`,`log10`,`sqrt`,`cos`,`tan`,`sin`... s'appliquent toutes terme-à-terme.

```
log10(c(10,100,1000))
```

```
## [1] 1 2 3
```

```
cos(c(pi/2,pi))^2 + sin(c(pi/2,pi))^2
```

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

Opérateurs mathématiques IV

Fonctions caractérisant un vecteur

`prod, sum, max, min, range, which.min, which.max, length`

```
x <- c(-8,1.5,3)
max(x)

## [1] 3

min(x)

## [1] -8

length(x)

## [1] 3
```

Opérateurs mathématiques V

```
prod(x)
```

```
## [1] -36
```

```
sum(x)
```

```
## [1] -3.5
```

```
range(x)
```

```
## [1] -8  3
```

```
which.max(x)
```

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

```
which.min(x)
```

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

Opérateurs mathématiques VI

Pour le minimum / maximum terme-à-terme

`pmin,pmax.`

```
x<-c(1,2,-3,-4); y<-c(-5,-6,9,0)
pmin(x,y)

## [1] -5 -6 -3 -4

pmax(x,y)

## [1] 1 2 9 0
```

Opérateurs mathématiques VII

Fonctions appliquées le long du vecteur

`cumsum, cumprod, cummin, cummax`

Opérateurs mathématiques VIII

```
x <- c(-2,1,-3,2)
cumprod(x)
```

```
## [1] -2 -2  6 12
```

```
cumsum(x)
```

```
## [1] -2 -1 -4 -2
```

```
cummax(x)
```

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

```
cummin(x)
```

```
## [1] -2 -2 -3 -3
```

Opérateurs mathématiques IX

La fonction tabulate

`tabulate` compte le nombre d'occurence de chaque entier dans un vecteur d'entiers

```
x <- c(2,1,1,2, 5)
tabulate(x)

## [1] 2 2 0 0 1

tabulate(x,nbin=4)

## [1] 2 2 0 0

tabulate(x,nbin=10)

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

Opérateurs ensemblistes I

Fonctionnent pour tous les modes

`unique,intersect,union,setdiff,setequal,is.element`

```
unique(c("banane","citron","banane"))
```

```
## [1] "banane" "citron"
```

```
intersect(c("banane","citron"),c("orange","banane"))
```

```
## [1] "banane"
```

```
union(c("banane","citron"),c("orange","banane"))
```

```
## [1] "banane" "citron" "orange"
```

```
setequal(c("banane","citron"),c("orange","banane"))
```

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

Opérateurs ensemblistes II

```r
is.element(1,sample(c(1,2,3),2))
```

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

```r
setdiff(c("banane","citron"),c("banane"))
```

```
## [1] "citron"
```

```r
setdiff(c("banane"),c("banane","citron"))
```

```
## character(0)
```

Faire une recherche dans un vecteur

Fonctionnent pour tous les modes

`match`, `%in%` (voir aussi `pmatch`...)

```
x <- rep(1:5,2); y <- c(3,5,1)
y %in% x

## [1] TRUE TRUE TRUE

x %in% y

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

match(x,y)

##  [1]  3 NA  1 NA  2  3 NA  1 NA  2

match(y,x)

## [1] 3 5 1
```

Tester le mode / les éléments d'un vecteur

Fonctionnent pour tous les modes

`is.xx,is.NA,is.infinite,all.equal,anyNA ...`

```
all.equal(c(1,12,3),c(1,12,3))

## [1] TRUE

anyNA(c(1,NA,3.5))

## [1] TRUE

is.numeric("allo?")

## [1] FALSE

!is.infinite(c(-Inf,12.5,7))

## [1] FALSE  TRUE  TRUE
```

Nommer les élément d'un vecteur

La fonction `names` permet d'accéder ou d'attribuer des noms aux éléments d'un vecteur.

```
x <- c(1,2,3)
names(x) <- c("one","two","three")
names(x)


## [1] "one"   "two"   "three"
```

La fonction `setNames` permet de créer directement un vecteur en nommant les éléments.

```
x <-setNames(c(1,2,3),c("one","two","three"))
x


##   one   two three
##     1     2     3
```

`names` confère un attribut à un objet vecteur

```
attributes(x)


## $names
## [1] "one"   "two"   "three"
```

Plan

Variables et types élémentaires

Vecteurs
- Définition
- Opérations élémentaires
- Génération de vecteurs
- Manipulation
- Représentation graphique minimal

Facteurs

Matrices (et tableaux)

Listes

Tableau de données

L'opérateur ':'

Génère une séquence de mode `numeric` par pas de un depuis le nombre `from` jusqu'à `to` (si possible).

```
-5:5
```

```
##  [1] -5 -4 -3 -2 -1  0  1  2  3  4  5
```

```
5:-5
```

```
##  [1]  5  4  3  2  1  0 -1 -2 -3 -4 -5
```

```
pi:6
```

```
## [1] 3.141593 4.141593 5.141593
```

```
1:6/2
```

```
## [1] 0.5 1.0 1.5 2.0 2.5 3.0
```

```
1:(6/2)
```

```
## [1] 1 2 3
```

La commande `seq`

Génère une séquence de mode `numeric`

Plusieurs schémas possibles

- `seq(from,to)`
- `seq(from,to,by=)`
- `seq(from,to,length.out=)`

```
seq(1,10)

##  [1]  1  2  3  4  5  6  7  8  9 10

seq(2,10,by=2)

## [1]  2  4  6  8 10

seq(2,10,length.out=6)

## [1]  2.0  3.6  5.2  6.8  8.4 10.0
```

Dérivées de seq

Quelques variantes

- `seq_along`
- `seq_len`
- `seq.int(from,to,length.out=)`

```
seq_len(7)

## [1] 1 2 3 4 5 6 7

seq.int(4,13,len=10)

##  [1]  4  5  6  7  8  9 10 11 12 13

seq_along(5:10)

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

La commande `rep`

Fonctionne pour tous les modes

- `rep(x,times)`, où `times` peut être un vecteur,
- `rep(x,each)`.
- `rep(x,length)`.

```
rep(1,3)

## [1] 1 1 1

rep(c("A","B","C"),c(3,2,4))

## [1] "A" "A" "A" "B" "B" "C" "C" "C" "C"

rep(c(TRUE,FALSE), each=2)

## [1]  TRUE  TRUE FALSE FALSE

rep(c(TRUE,FALSE), length=5)

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

Dérivées de `rep`

Quelques variantes

Plus simples, plus rapides, mais oublie les attributs du vecteur

- `rep.int(x,times),,`
- `rep_len(x,length_out).`

```
x <- setNames(rep("Mercy",3), c("one","two","three"))
rep(x,c(1,2,3))

##     one     two     two   three   three   three
## "Mercy" "Mercy" "Mercy" "Mercy" "Mercy" "Mercy"

rep.int(x,c(1,2,3))

## [1] "Mercy" "Mercy" "Mercy" "Mercy" "Mercy" "Mercy"

rep_len(x,7)

## [1] "Mercy" "Mercy" "Mercy" "Mercy" "Mercy" "Mercy" "Mercy"
```

Génération de vecteurs logiques

Obtenus par conditions avec

- les opérateurs logiques '<', '<=', '>', '>=', '==' '!='
- le ET, le OU, NON, OU exclusif : '&' (intersection), '|' (union), '!' (négation), `xor`.

```
note1 <- c(8,9,14,3,17.5,11)
note2 <- c("C","B","A","B","E","B")
admis <- (note1 >= 10) & (note2 == "A" | note2 == "B")
mention <- (note1 >= 15) & (note2 == "A")
admis
```

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

```
sum(admis)
```

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

```
sum(mention)
```

```
## [1] 0
```

Par concaténation

Avec `'c()'`

L'opérateur `'c()'` peut s'appliquer à n'importe quoi pourvu que l'on concatène des vecteurs de même type.

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

## [1] 1 2 3 4

round(c(seq(-pi,pi,len=4),rep(c(1:3),each=2),0),2)

##  [1] -3.14 -1.05  1.05  3.14  1.00  1.00  2.00  2.00  3.00  3.00  0.00
```

Remarque

Dans le second exemple, les entiers composants `c(1:3)` ont été forcés au typage flottant.

Par concaténation II

Avec `paste`

Concaténation de chaînes de caractères. Convertit en caractères les éléments passés en argument avant toute opération.

```
paste("R","c'est","bien")

## [1] "R c'est bien"

paste(2:4,"ieme")

## [1] "2 ieme" "3 ieme" "4 ieme"

paste("A",1:5, sep="")

## [1] "A1" "A2" "A3" "A4" "A5"

paste("A",1:5, sep="",collapse="")

## [1] "A1A2A3A4A5"
```

Plan

Variables et types élémentaires

Vecteurs
- Définition
- Opérations élémentaires
- Génération de vecteurs
- Manipulation
- Représentation graphique minimal

Facteurs

Matrices (et tableaux)

Listes

Tableau de données

Indexation des vecteurs

Extrêmement puissant !

Principe

- Permet la sélection d'un sous-ensemble du vecteur `x`.
- Permet également d'affecter de nouvelles valeurs.
- Le sous-ensemble est spécifié entre crochets `x[subset]`.

L'objet `subset` peut prendre 4 types différents :

1. un vecteur logique, qui doit être de la même taille que le vecteur `x` ;
2. un vecteur numérique aux composantes positives, qui spécifie les valeurs à inclure ;
3. un vecteur numérique aux composantes négatives, qui spécifie les valeurs à exclure ;
4. un vecteur de chaînes de caractères, qui spécifie les noms des éléments de `x` à conserver.

Indexation des vecteurs

Extrêmement puissant !

Principe

- Permet la sélection d'un sous-ensemble du vecteur `x`.
- Permet également d'affecter de nouvelles valeurs.
- Le sous-ensemble est spécifié entre crochets `x[subset]`.

L'objet `subset` peut prendre 4 types différents :

1. un vecteur logique, qui doit être de la même taille que le vecteur `x` ;
2. un vecteur numérique aux composantes positives, qui spécifie les valeurs à inclure ;
3. un vecteur numérique aux composantes négatives, qui spécifie les valeurs à exclure ;
4. un vecteur de chaînes de caractères, qui spécifie les noms des éléments de `x` à conserver.

Indexation des vecteurs

Extrêmement puissant !

Principe

- Permet la sélection d'un sous-ensemble du vecteur `x`.
- Permet également d'affecter de nouvelles valeurs.
- Le sous-ensemble est spécifié entre crochets `x[subset]`.

L'objet `subset` peut prendre 4 types différents :

1. un vecteur logique, qui doit être de la même taille que le vecteur `x` ;
2. un vecteur numérique aux composantes positives, qui spécifie les valeurs à inclure ;
3. un vecteur numérique aux composantes négatives, qui spécifie les valeurs à exclure ;
4. un vecteur de chaînes de caractères, qui spécifie les noms des éléments de `x` à conserver.

Indexation des vecteurs

Extrêmement puissant !

Principe

- Permet la sélection d'un sous-ensemble du vecteur `x`.
- Permet également d'affecter de nouvelles valeurs.
- Le sous-ensemble est spécifié entre crochets `x[subset]`.

L'objet `subset` peut prendre 4 types différents :

1. un vecteur logique, qui doit être de la même taille que le vecteur `x` ;
2. un vecteur numérique aux composantes positives, qui spécifie les valeurs à inclure ;
3. un vecteur numérique aux composantes négatives, qui spécifie les valeurs à exclure ;
4. un vecteur de chaînes de caractères, qui spécifie les noms des éléments de `x` à conserver.

Indexation des vecteurs

Extrêmement puissant !

Principe

- Permet la sélection d'un sous-ensemble du vecteur `x`.
- Permet également d'affecter de nouvelles valeurs.
- Le sous-ensemble est spécifié entre crochets `x[subset]`.

L'objet `subset` peut prendre 4 types différents :

1. un vecteur logique, qui doit être de la même taille que le vecteur `x` ;
2. un vecteur numérique aux composantes positives, qui spécifie les valeurs à inclure ;
3. un vecteur numérique aux composantes négatives, qui spécifie les valeurs à exclure ;
4. un vecteur de chaînes de caractères, qui spécifie les noms des éléments de `x` à conserver.

Indexation des vecteurs : exemples I

Vecteurs logiques

```
x <- c(3,6,-2,9,NA,sin(-pi/6))
x[x > 0]

## [1]  3  6  9 NA

x[!is.na(x)]

## [1]  3.0  6.0 -2.0  9.0 -0.5

x[!is.na(x) & x>0]

## [1] 3 6 9

x[x <= mean(x,na.rm=TRUE)]

## [1]  3.0 -2.0   NA -0.5
```

Indexation des vecteurs : exemples II

Vecteurs aux composantes positives ou négatives

```
x[2]

## [1] 6

x[1:5]

## [1]  3  6 -2  9 NA

x[-c(1,5)]

## [1]  6.0 -2.0  9.0 -0.5

x[-(1:5)]

## [1] -0.5
```

Indexation des vecteurs : exemples III

Vecteurs de chaînes de caractères

```
names(x) <- c("var1","var2","var3","var4","var5","var6")
x

## var1 var2 var3 var4 var5 var6
##  3.0  6.0 -2.0  9.0   NA -0.5

x[c("var1","var3")]

## var1 var3
##    3   -2
```

Autres commandes d'indexation et de sélection

1. Classer
   - `sort` renvoie le vecteur classé par ordre croissant ou décroissant,
   - `order` renvoie les indices d'ordre des éléments par ordre croissant ou décroissant,
2. Extraire
   - `which` renvoie les indices de `x` vérifiant une condition ;
3. Échantillonner
   - `sample` échantillonne aléatoirement dans un vecteur `x`, avec ou sans remise.

Autres commandes d'indexation et de sélection

1. Classer
   - `sort` renvoie le vecteur classé par ordre croissant ou décroissant,
   - `order` renvoie les indices d'ordre des éléments par ordre croissant ou décroissant,

2. Extraire
   - `which` renvoie les indices de `x` vérifiant une condition ;

3. Échantillonner
   - `sample` échantillonne aléatoirement dans un vecteur `x`, avec ou sans remise.

Autres commandes d'indexation et de sélection

1. Classer
   - `sort` renvoie le vecteur classé par ordre croissant ou décroissant,
   - `order` renvoie les indices d'ordre des éléments par ordre croissant ou décroissant,

2. Extraire
   - `which` renvoie les indices de `x` vérifiant une condition ;

3. Échantillonner
   - `sample` échantillonne aléatoirement dans un vecteur `x`, avec ou sans remise.

Exemples

```
x <- -5:5
y <- sample(x)
sort(y)
```

```
##  [1] -5 -4 -3 -2 -1  0  1  2  3  4  5
```

```
order(y)
```

```
##  [1]  2 10  8  6 11  5  3  4  9  7  1
```

```
y[order(y)]
```

```
##  [1] -5 -4 -3 -2 -1  0  1  2  3  4  5
```

```
y[order(y,decreasing=TRUE)]
```

```
##  [1]  5  4  3  2  1  0 -1 -2 -3 -4 -5
```

```
which(sample(x,4) > 0)
```

```
## [1] 3 4
```

Fonctions statistiques élémentaire

moyenne, médiance variance, écart-type :

```
x <- -2:5
mean(x)

## [1] 1.5

median(x)

## [1] 1.5

var(x)

## [1] 6

sd(x)

## [1] 2.44949
```

Plan

Variables et types élémentaires

Vecteurs
- Définition
- Opérations élémentaires
- Génération de vecteurs
- Manipulation
- Représentation graphique minimal

Facteurs

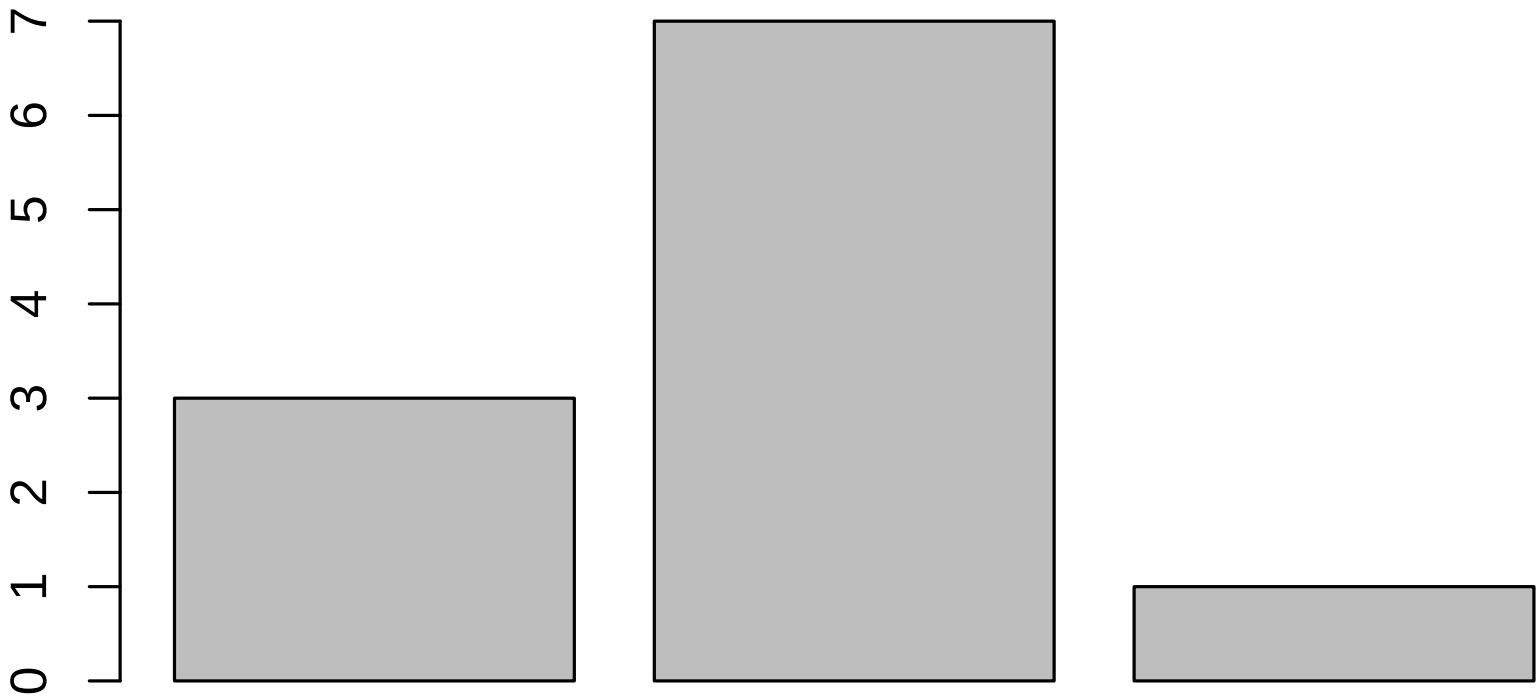
Matrices (et tableaux)

Listes

Tableau de données

la fonction barplot

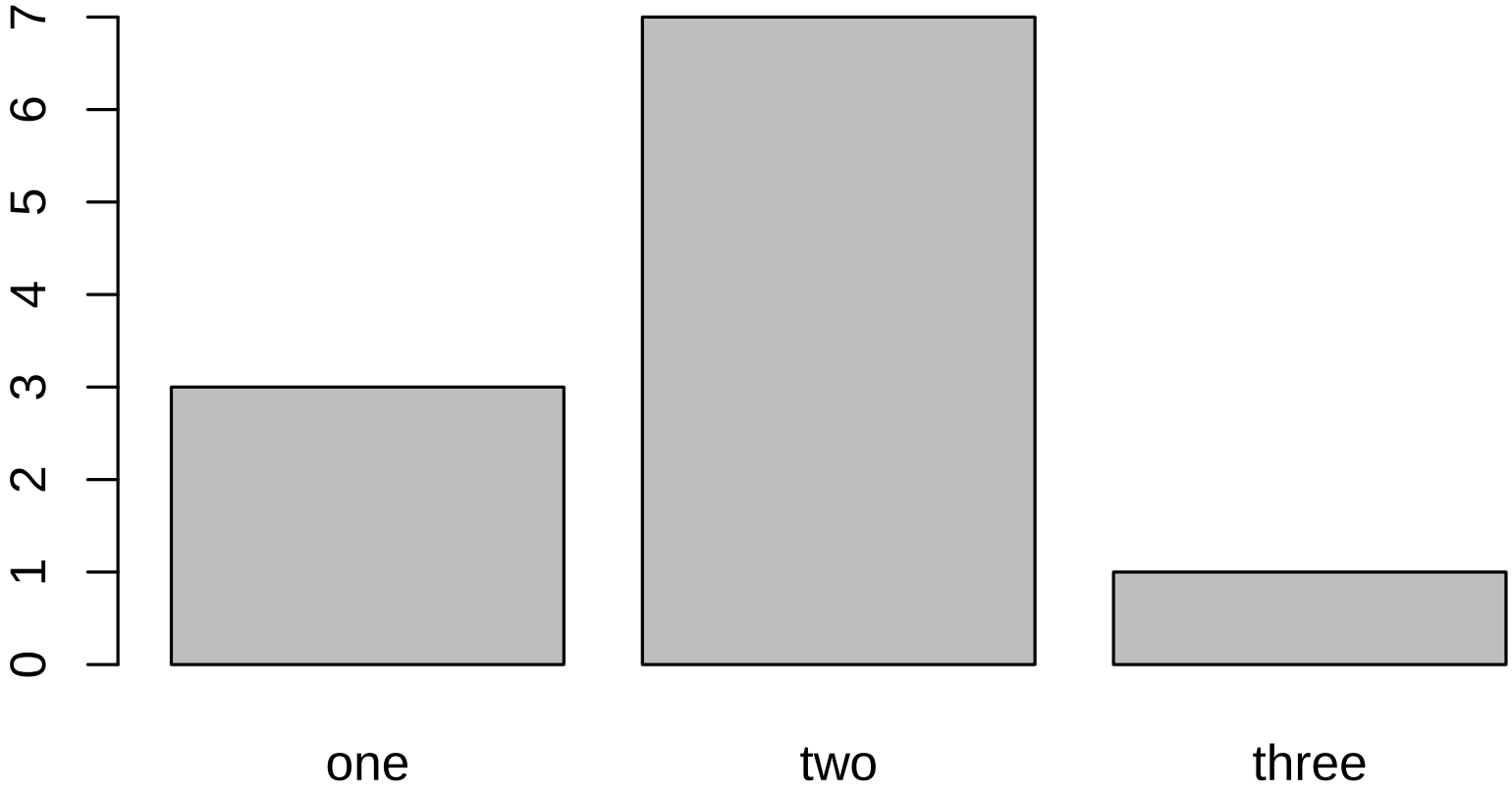
```
x <-c(3,7,1)
barplot(x)
```



la fonction barplot II

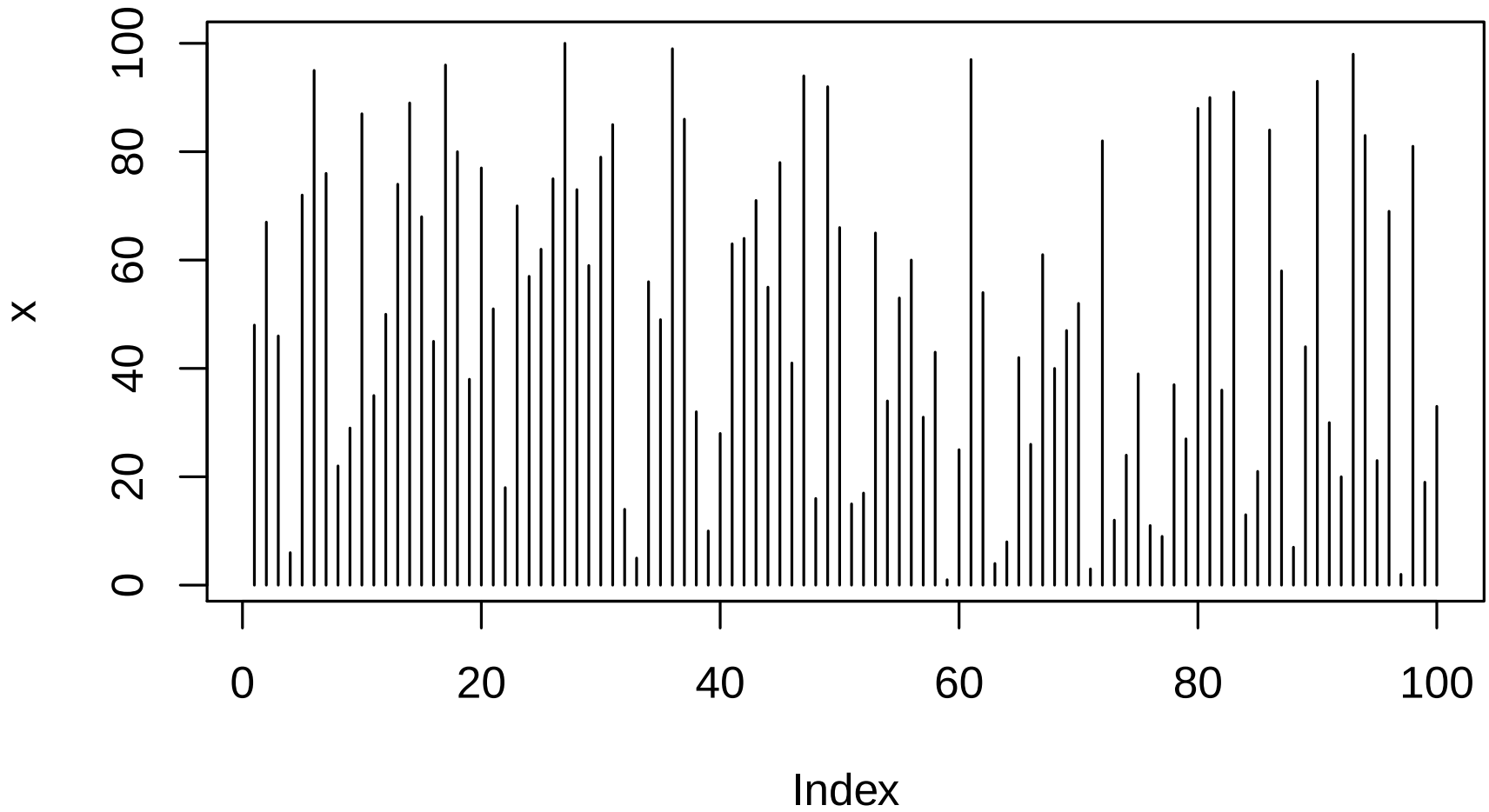
De l'utilité des attributs et du typage

```
x <-setNames(c(3,7,1),c("one","two","three"))
barplot(x)
```



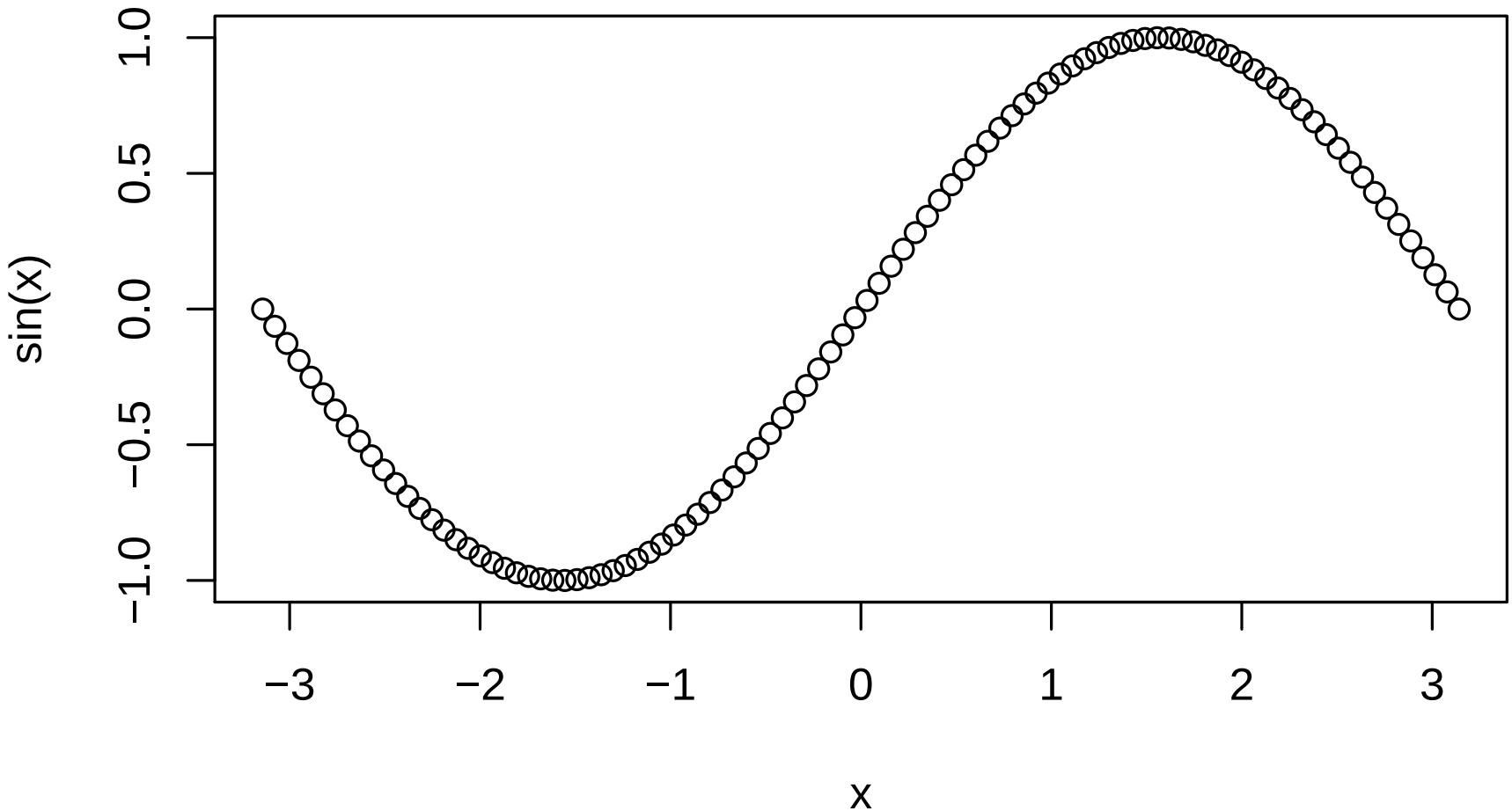
la fonction plot

```r
x <- sample(1:100)
plot(x, type="h")
```



la fonction plot II

```r
x <- seq(-pi,pi,len=100)
plot(x, sin(x))
```



Plan

Variables et types élémentaires

Vecteurs

Facteurs
- Définition
- Manipulation
- Utilisation
- Représentation graphique minimal

Matrices (et tableaux)

Listes

Tableau de données

Plan

Variables et types élémentaires

Vecteurs

Facteurs
- Définition
- Manipulation
- Utilisation
- Représentation graphique minimal

Matrices (et tableaux)

Listes

Tableau de données

Définition

Définition

Un facteur est un vecteur de variables catégorielles. Les niveaux du facteur peuvent être ordonnés ou pas.

Utilisation

les facteurs s'utilisent pour catégoriser les données d'un vecteur (ce qui s'avère très utile pour la gestion des variables qualitatives).

$\rightsquigarrow$ un facteur est souvent associé à d'autres vecteurs pour en définir une partition.

Définition

Définition

Un facteur est un vecteur de variables catégorielles. Les niveaux du facteur peuvent être ordonnés ou pas.

Utilisation

les facteurs s'utilisent pour catégoriser les données d'un vecteur (ce qui s'avère très utile pour la gestion des variables qualitatives).

$\rightsquigarrow$ un facteur est souvent associé à d'autres vecteurs pour en définir une partition.

Définition

Définition

Un facteur est un vecteur de variables catégorielles. Les niveaux du facteur peuvent être ordonnés ou pas.

Utilisation

les facteurs s'utilisent pour catégoriser les données d'un vecteur (ce qui s'avère très utile pour la gestion des variables qualitatives).

$\rightsquigarrow$ un facteur est souvent associé à d'autres vecteurs pour en définir une partition.

Création

Création : la fonction `factor`

```r
x <- sample(1:3,10,replace=TRUE)
factor(x)
```

```
##  [1] 3 2 1 3 3 2 3 1 1 2
## Levels: 1 2 3
```

```r
as.factor(x)
```

```
##  [1] 3 2 1 3 3 2 3 1 1 2
## Levels: 1 2 3
```

```r
factor(x,levels=1:5)
```

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

```r
factor(x,levels=1:5, ordered=TRUE)
```

```
##  [1] 3 2 1 3 3 2 3 1 1 2
## Levels: 1 < 2 < 3 < 4 < 5
```

Le type d'objet facteur

Un facteur est en fait un vecteur d'entier muni d'un attribut `levels` :

```
x
```

```
##  [1] 3 2 1 3 3 2 3 1 1 2
```

```
fx <- factor(x)
attributes(fx)
```

```
## $levels
## [1] "1" "2" "3"
##
## $class
## [1] "factor"
```

```
as.numeric(fx)
```

```
##  [1] 3 2 1 3 3 2 3 1 1 2
```

Plan

Variables et types élémentaires

Vecteurs

Facteurs
- Définition
- Manipulation
- Utilisation
- Représentation graphique minimal

Matrices (et tableaux)

Listes

Tableau de données

Accès/affectation des attributs I

Gestion : `nlevels,levels,table`

```
levels <- c("MdC","thésard","CR")
data <- sample(levels,15,replace=TRUE)
fx  <- factor(data)
nlevels(fx)

## [1] 3

levels(fx)

## [1] "CR"      "MdC"     "thésard"

table(fx)

## fx
##      CR     MdC thésard
##       2       2      11
```

Accès/affectation des attributs II

```
levels(fx) <- c("Dr","CR","MdC","thésards")
nlevels(fx)

## [1] 4

levels(fx)

## [1] "Dr"       "CR"       "MdC"      "thésards"

table(fx)

## fx
##       Dr       CR      MdC thésards
##        2        2       11        0

is.ordered(fx)

## [1] FALSE
```

Accès/affectation des attributs III

```r
fx <- factor(data, levels=c("thésard","MdC","CR","Dr"), ordered=TRUE)
nlevels(fx)
```

```
## [1] 4
```

```r
levels(fx)
```

```
## [1] "thésard" "MdC"     "CR"      "Dr"
```

```r
table(fx)
```

```
## fx
## thésard     MdC      CR      Dr
##      11       2       2       0
```

```r
is.ordered(fx)
```

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

Plan

Variables et types élémentaires

Vecteurs

Facteurs
- Définition
- Manipulation
- Utilisation
- Représentation graphique minimal

Matrices (et tableaux)

Listes

Tableau de données

Facteur associé à un vecteur

Permet de manipuler un vecteur conditionnellement à un facteur

Données

Dans le laboratoire, chacun me donne son âge et son grade [1]

```
age <- c(25,35,32,27,32,40,26,25,26,28,30,NA,36,30,30)
grd <- c("thd","CR","MdC","thd","thd","MdC","MdC","thd","thd","MdC","CR","MdC","CR"
```

Question : nombre d'individus par catégorie ?

```
table(grd)

## grd
##  CR MdC thd
##   3   5   7
```

1. sauf un qui refuse :'(

La fonction split

Découpe un vecteur selon les valeurs d'un facteur

```
split(age,grd)


## $CR
## [1] 35 30 36
##
## $MdC
## [1] 32 40 26 28 NA
##
## $thd
## [1] 25 27 32 25 26 30 30
```

La fonction `tapply`

Utilisation

Applique une fonction sur un vecteur partitionné en groupes. Généralise l'exemple de la fonction `split`, en appliquant n'importe quelle fonction au résultat obtenu

Question : âge moyen / écart-type par catégorie ?

```
tapply(age,grd,mean,na.rm=TRUE)

##       CR      MdC      thd
## 33.66667 31.50000 27.85714

tapply(age,grd,sd,na.rm=TRUE)

##       CR      MdC      thd
## 3.214550 6.191392 2.794553
```

La fonction `rowsum`

Équivalent de `tapply`, `index`, `sum` amis en beaucoup plus rapide

```
tapply(age,grd,sum,na.rm=TRUE)

##  CR MdC thd
## 101 126 195

rowsum(age,grd,na.rm=TRUE)

##     [,1]
## CR   101
## MdC  126
## thd  195
```

La fonction cut

Découpe un vecteur selon des ruptures et les distribue par intervalle

```
cut(2:8,c(0,5,10))
```

```
## [1] (0,5]  (0,5]  (0,5]  (0,5]  (5,10] (5,10] (5,10]
## Levels: (0,5] (5,10]
```

Plan

Variables et types élémentaires

Vecteurs

Facteurs
- Définition
- Manipulation
- Utilisation
- Représentation graphique minimal

Matrices (et tableaux)

Listes

Tableau de données

la fonction barplot

Aïe !

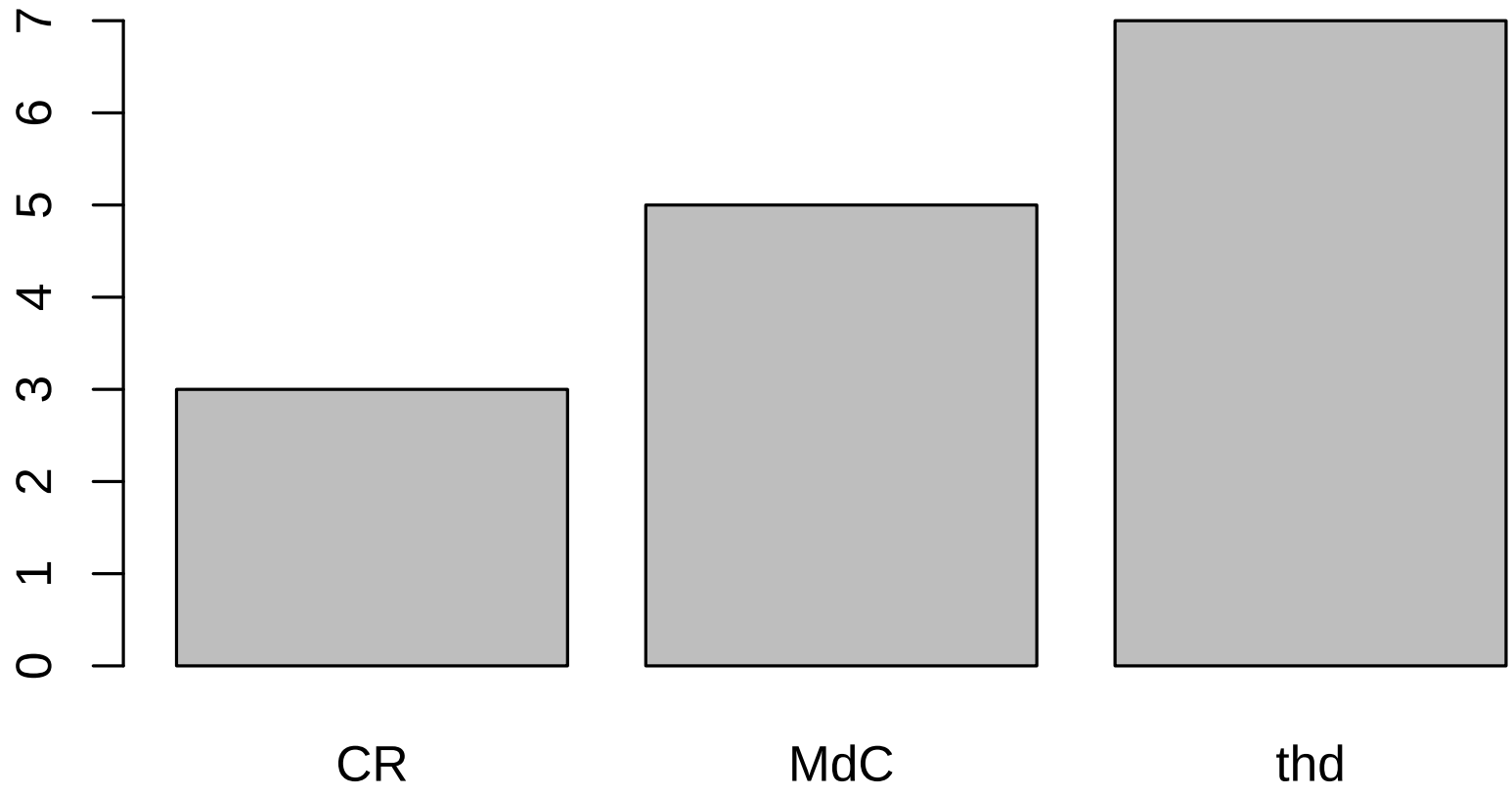
```
plot(grd)
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): NAs introduits lors de la
conversion automatique
## Warning in min(x): aucun argument trouvé pour min ; Inf est renvoyé
## Warning in max(x): aucun argument pour max ; -Inf est renvoyé
## Error in plot.window(...): valeurs finies requises pour 'ylim'
```

la fonction barplot II

De l'utilité des attributs et du typage

```
plot(factor(grd))
```



Plan

Variables et types élémentaires

Vecteurs

Facteurs

Matrices (et tableaux)
- Définition
- Manipulation
- Opérateurs matriciels
- Représentation graphique des matrices

Listes

Tableau de données

Plan

Variables et types élémentaires

Vecteurs

Facteurs

Matrices (et tableaux)
- Définition
- Manipulation
- Opérateurs matriciels
- Représentation graphique des matrices

Listes

Tableau de données

Tableau : définition

Définition (objet `array`)

Un tableau est un vecteur muni d'un attribut dimension (`dim`), lui même défini par un vecteur. Il est défini par la commande `array(data,dim,dimnames=)`

```
array(1:8,c(2,2,2))
```

```
## , , 1
##
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## , , 2
##
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
```

Matrice : définition

Définition (objet `matrix`)

Une matrice est un tableau à deux dimensions. Elle est définie par la commande

`matrix(data,nrow=,ncol=,byrow)`

En conséquence

- Un objet `array` à deux dimensions est automatiquement converti en `matrix`
- Un vecteur auquel on ajoute un attribut dimension est automatiquement converti en `matrix`

```
class(array(1:4,c(2,2)))

## [1] "matrix"

x <- c(1,2,3,4)
dim(x) <- c(2,2)
class(x)

## [1] "matrix"
```

Matrice et tableau : attributs dimension

On accède aux attributs de dimension d'un tableau à l'aide des commandes `dim` et `nrow`,`ncol` dans le cas d'une matrice.

```
A <- array(1:12,c(2,2,3))
M <- matrix(1:6,c(2,3))
dim(A)

## [1] 2 2 3

dim(M)

## [1] 2 3

nrow(M)

## [1] 2

ncol(M)

## [1] 3
```

Remarques importantes

1. R range les éléments d'une matrice par défaut par colonne.

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

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

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

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

2. Lors de la création d'une matrice, R recycle les éléments jusqu'à ce que les contraintes de dimension soient vérifiées.

```
matrix(1:3,nrow=2,ncol=2)
```

```
## Warning in matrix(1:3, nrow = 2, ncol = 2): la longueur des données [3]
n'est pas un diviseur ni un multiple du nombre de lignes [2]
```

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

Plan

Variables et types élémentaires

Vecteurs

Facteurs

Matrices (et tableaux)
- Définition
- Manipulation
- Opérateurs matriciels
- Représentation graphique des matrices

Listes

Tableau de données

Matrices : opérateurs élémentaires

Étant donné qu'une matrice est un vecteur pourvu d'une dimension, on a la proposition suivante :

Proposition

La plupart des opérateurs vectorielles s'appliquent (arithmétiques/mathématiques, ensemblistes, d'indexation).

```
a <- matrix(sample(-4:4,9),3,3)
cat(max(a),sum(a),prod(a))

## 4 0 0

which(a > 0)

## [1] 4 7 8 9

cumsum(a[a > 0])

## [1]  3  4  6 10

order(a)
```

Manipulation de matrices

Opérateurs matriciels usuels

- `+,/,*,^` sont les opérateurs usuels terme-à-terme,
- `%*%` est le produit matriciel,
- `crossprod()` est le produit scalaire,
- `t()` transpose une matrice,
- `diag()` extrait / spécifie la diagonale.

```
a <- matrix(sample(-4:4,9),3,3)
b <- matrix(sample(a),3,3)
diag(a)

## [1]  4 -2 -4

diag(a) <- diag(b) <- 1
diag(a)

## [1] 1 1 1
```

Indexation propres aux matrices

En ligne et/ou en colonne

Selon les mêmes techniques que pour un vecteur

```
a[1:2,]
```

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

```
a[,-3]
```

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

```
a[-2,c(3,1)]
```

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

Indexation propres aux matrices II

À l'aide d'une matrice de booléén

Nécessite la création d'une matrice appropriée

```
upper.tri(a)

##       [,1]  [,2]  [,3]
## [1,] FALSE  TRUE  TRUE
## [2,] FALSE FALSE  TRUE
## [3,] FALSE FALSE FALSE

a[upper.tri(a)]

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

Indexation propres aux matrices III

À l'aide des noms de ligne et de colonnes

Nécessite d'avoir des attributs `colnames/rownames`

```r
colnames(a) <- paste0("c",as.character(1:3))
rownames(a) <- paste0("r",as.character(1:3))
a
```

```
##    c1 c2 c3
## r1  1 -3  1
## r2 -1  1  3
## r3  0  2  1
```

```r
a[rownames(a) == "r2", colnames(a) %in% c("c3","c2")]
```

```
## c2 c3
##  1  3
```

Concaténation de matrices I

Trois fonctions selon l'effet voulu :

1. `c()` concatène les éléments en un vecteur,
2. `cbind()` empile horizontalement plusieurs matrices,
3. `rbind()` empile verticalement plusieurs matrices.

```
a <- matrix(1,2,3)
b <- matrix(2,2,3)
c(a,b)

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

Concaténation de matrices II

```
cbind(a,b)
```

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

```
rbind(a,b)
```

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

Plan

Variables et types élémentaires

Vecteurs

Facteurs

Matrices (et tableaux)
- Définition
- Manipulation
- Opérateurs matriciels
- Représentation graphique des matrices

Listes

Tableau de données

Quelques fonctions spécifiques

Opération en ligne/en colonne

Les commandes `colSums`,`rowSums`,`colMeans`,`rowMeans`

```
colSums(a)
```

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

```
rowSums(a)
```

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

```
rowMeans(a)
```

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

```
colMeans(a)
```

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

Plus généralement et pour les tableau : `apply`

Opération en ligne/en colonne

Les commandes `apply(array, dim, function)`

```
apply(a, 2, max)

## [1] 1 1 1

a <- array(1:12, c(2,3,2))
apply(a, 3, colMeans)

##      [,1] [,2]
## [1,]  1.5  7.5
## [2,]  3.5  9.5
## [3,]  5.5 11.5
```

$\rightsquigarrow$ Très puissant !

Algèbre linéaire élémentaire

Résolution de systèmes linéaires, inversion matricielle

La commande `solve` résout

$$\mathbf{A}x = \mathbf{b},$$

```
A <- matrix(c(4,2,8,-3),2,2)
b <- c(2,3)
solve(A,b)
```

```
## [1]  1.0714286 -0.2857143
```

ou inverse une matrice :

```
round(solve(A) %*% A,8)
```

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

Commandes avancées d'algèbre linéaire

Utile pour l'analyse numérique

`R` dispose des outils classiques d'algèbre linéaire

- `det` : calcule le déterminant d'une matrice ;
- `chol` : factorisation de Cholesky ($A = C^\intercal C$, avec A symétrique, C triangulaire supérieure) ;
- `qr` : factorisation QR ($A = QR$ avec Q orthogonale, R triangulaire supérieure) ;
- `eigen` : calcule valeurs propres et vecteurs propres d'une matrice ;
- `svd` : calcule la décomposition en valeurs singulières.
- `...`

Plan

Variables et types élémentaires

Vecteurs

Facteurs

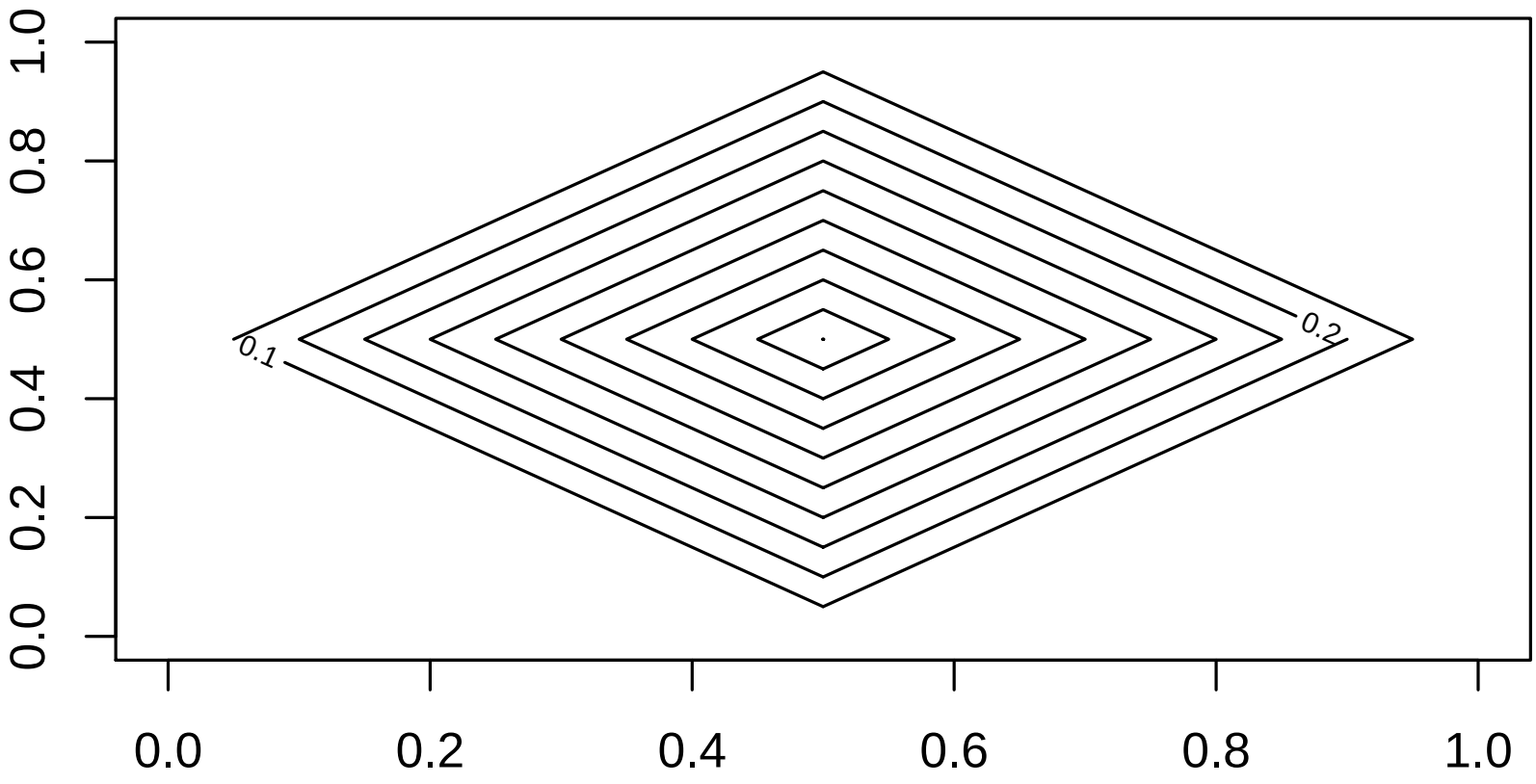
Matrices (et tableaux)
- Définition
- Manipulation
- Opérateurs matriciels
- Représentation graphique des matrices

Listes

Tableau de données

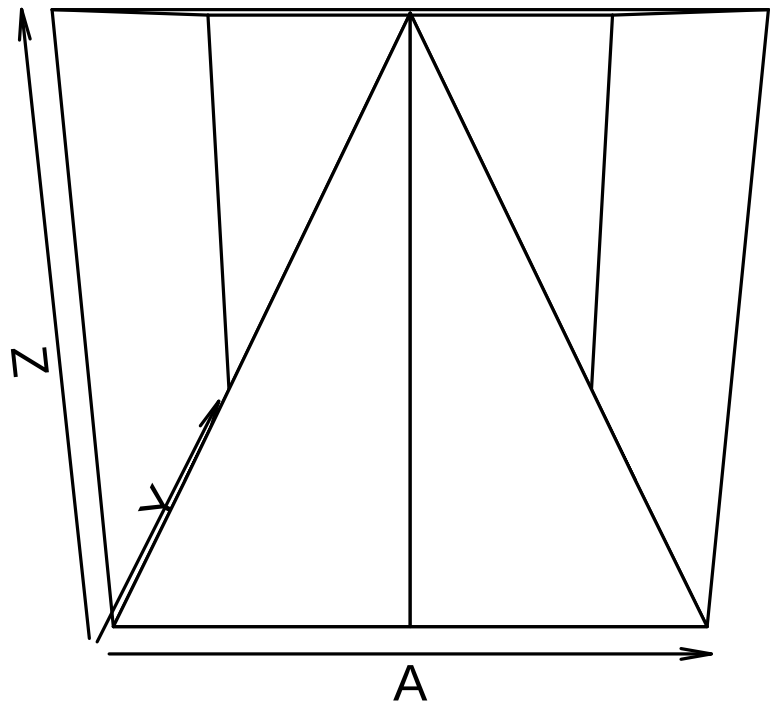
Représentation tridimensionnelle

```
A <- matrix(0,3,3); A[2,2] <- 1
contour(A)
```



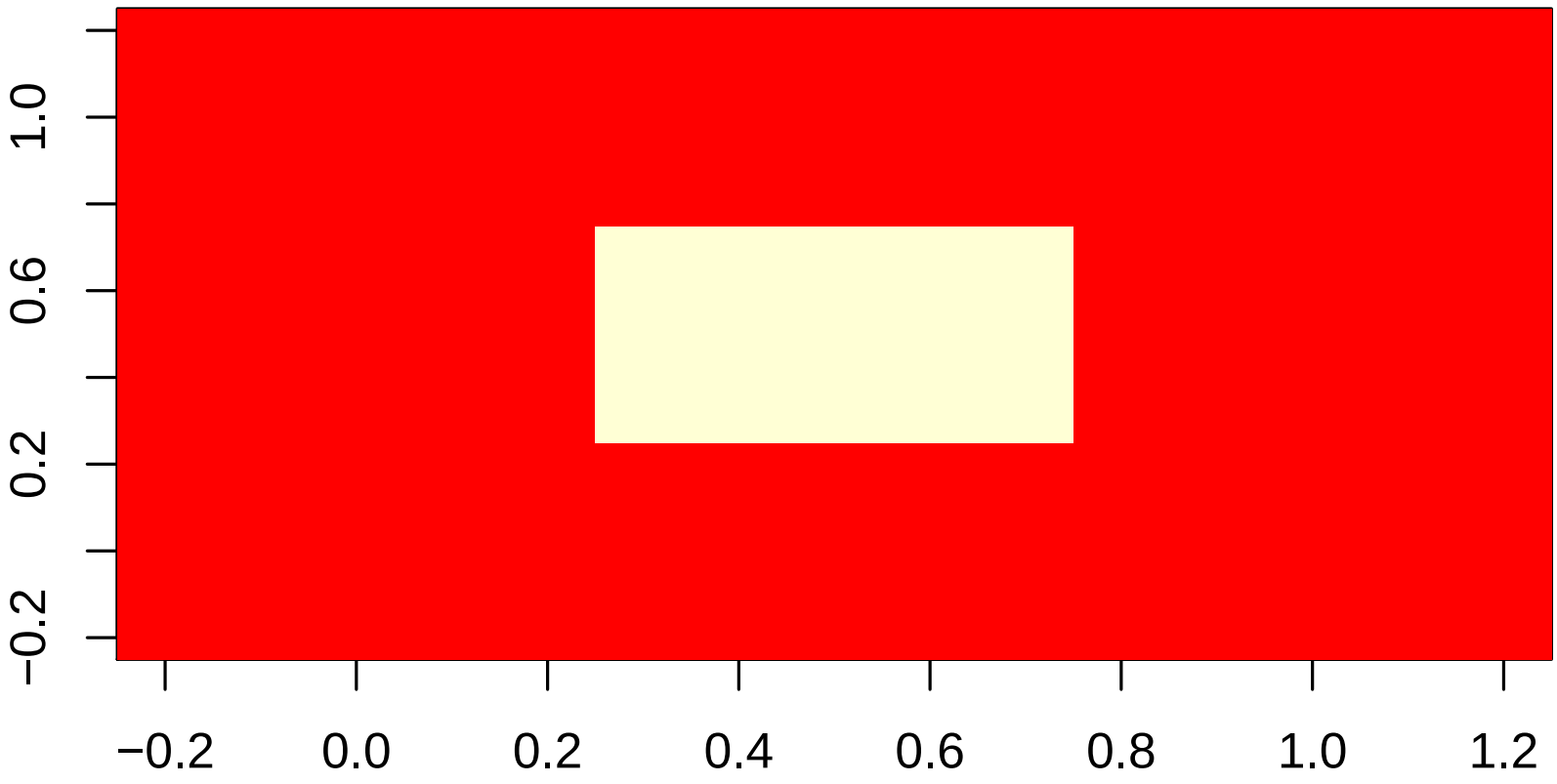
Représentation tridimensionnelle II

```
persp(A)
```



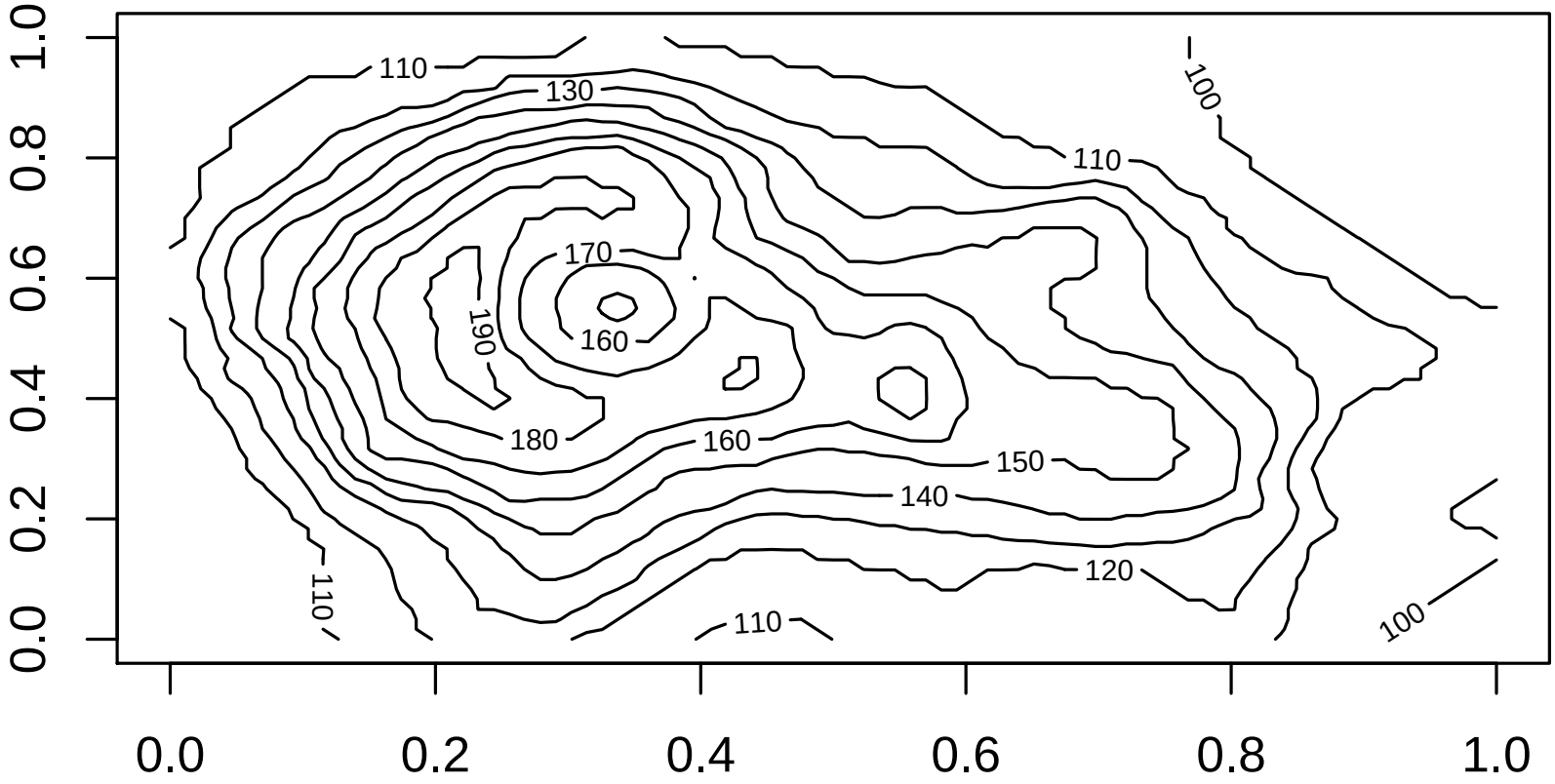
Représentation tridimensionnelle III

```
image(A)
```



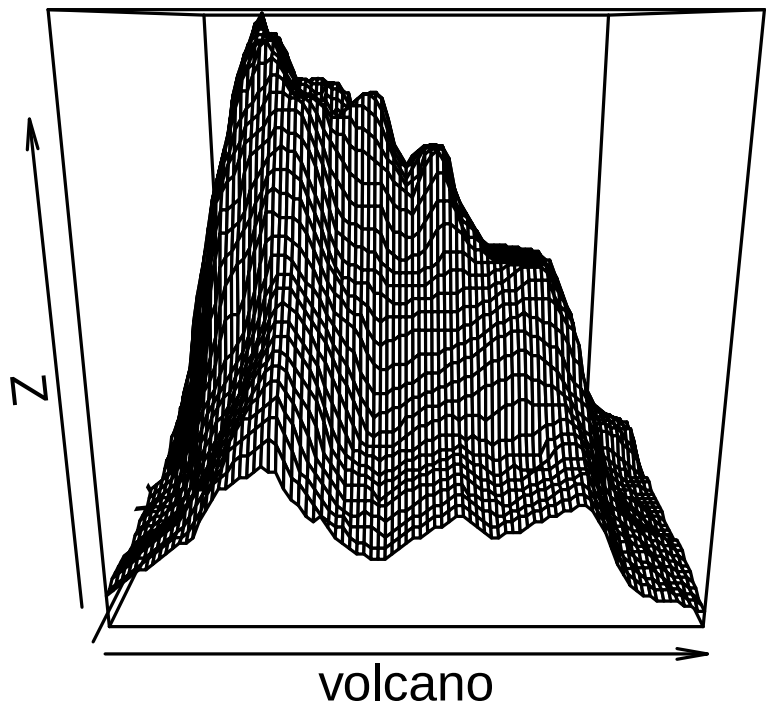
Représentation tridimensionnelle IV

```
contour(volcano)
```



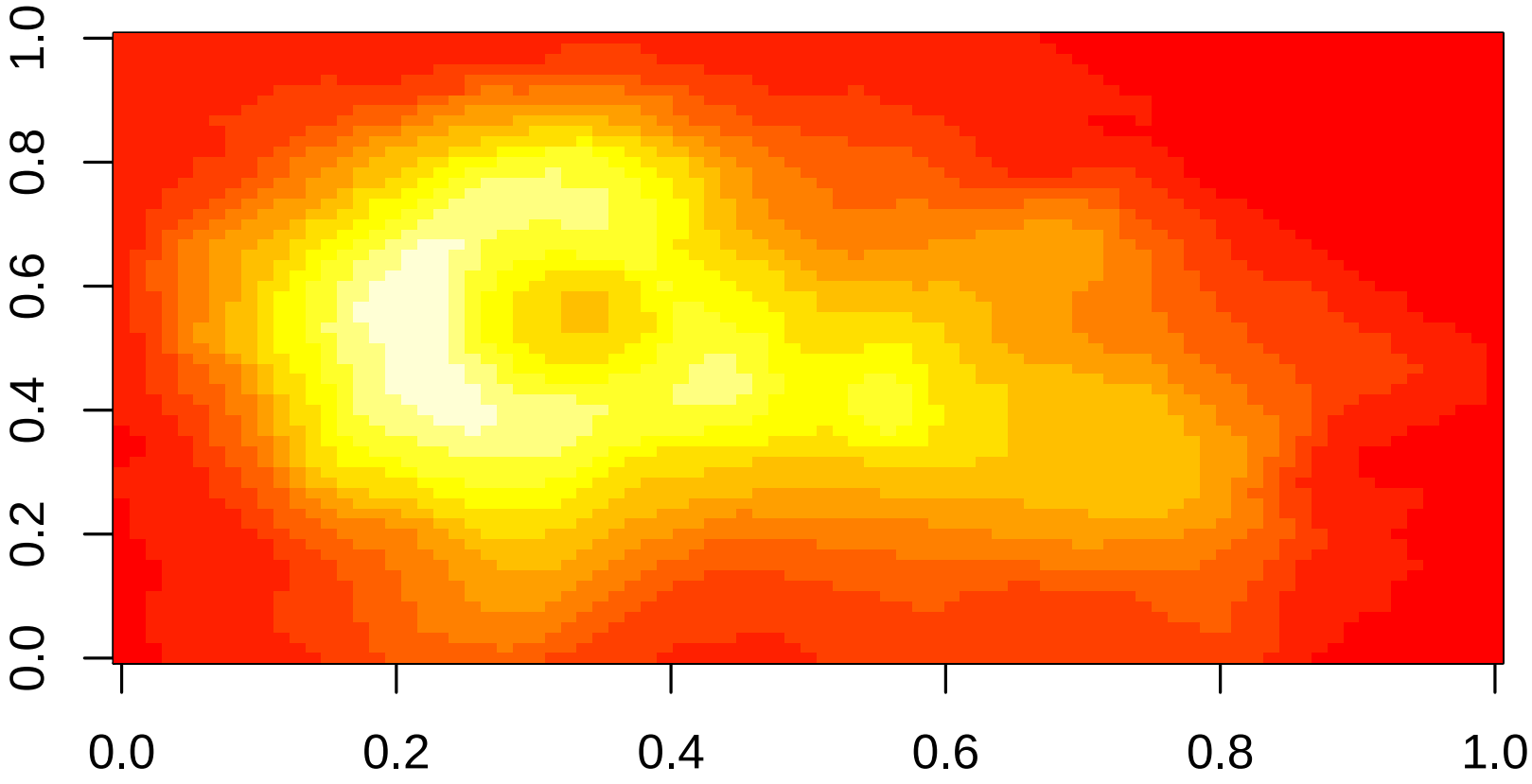
Représentation tridimensionnelle V

```
persp(volcano)
```



Représentation tridimensionnelle VI

```
image(volcano)
```



Plan

Variables et types élémentaires

Vecteurs

Facteurs

Matrices (et tableaux)

Listes
- Définition
- Manipulation

Tableau de données

Plan

Variables et types élémentaires

Vecteurs

Facteurs

Matrices (et tableaux)

Listes
- Définition
- Manipulation

Tableau de données

Liste I

Définition (objet `list`)

Une liste est une collection d'objets hétérogènes. Elle est définie par la commande `list(el1=, el2=, ...)`.

```
list(c(1,2,3),c("robert","johnson"),matrix(rnorm(4),2,2))

## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] "robert"  "johnson"
##
## [[3]]
##             [,1]       [,2]
## [1,] -0.4724784 -0.8086764
## [2,] -0.7466398  0.4368232
```

Liste II

Peut aussi être initialisée par

```
vector("list", 3)

## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
```

Liste III

Les éléments d'une liste peuvent posséder un nom à la définition ou être attribués *a posteriori*.

```
list(numero = c(1,2,3), noms = c("robert","johnson"), mat = matrix(rnorm(4),2,2))
```

```
## $numero
## [1] 1 2 3
##
## $noms
## [1] "robert"  "johnson"
##
## $mat
##           [,1]       [,2]
## [1,] 0.2288784 -0.4130860
## [2,] 0.4126223 -0.3230689
```

```
l <- list(c(1,2,3), c("robert","johnson"), matrix(rnorm(4),2,2))
names(l) <- c("numero","noms","mat")
```

Plan

Variables et types élémentaires

Vecteurs

Facteurs

Matrices (et tableaux)

Listes

- Définition
- Manipulation

Tableau de données

Accéder aux éléments d'une liste I

Les éléments de la liste ne sont pas nommés

On accède au i^{e} élément par indexation `nom_liste[[i]]` uniquement.

```
maliste[[2]]

## [1] "robert"  "johnson"

maliste[[2]][2]

## [1] "johnson"
```

Accéder aux éléments d'une liste II

Les éléments de la liste sont nommés

On peut accéder comme ci-dessus ou en utilisant le nom de l'élément `nom_liste$nom_elt`.

```
maliste <- list(numero = c(1,2,3), noms = c("robert","johnson"), mat = matrix(rnorm
maliste$nom

## [1] "robert"  "johnson"

maliste$nom[2]

## [1] "johnson"
```

Sélectionner des éléments I

Fonctionne (presque) comme pour les vecteurs

Attention à la différence `[[]]` et `[]`

```
l1 <- list(1:2,c("a","c","g","t"),matrix(NA,2,1))
l1[-c(1,3)]

## [[1]]
## [1] "a" "c" "g" "t"

mode(l1[3])

## [1] "list"

mode(l1[[3]])

## [1] "logical"
```

Dépiler une liste I

Commande `unlist`

mets à plat une liste et simplifie en vecteur si possible.

```
unlist(list(1,2:5))

## [1] 1 2 3 4 5

unlist(list("a",2:5))

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

Par defaut, récursivement

```
l <- list(list(1,"a",matrix(0,2,1)),c("b","c"),1:2)
unlist(l)

## [1] "1" "a" "0" "0" "b" "c" "1" "2"
```

Dépiler une liste II

```
unlist(l, recursive=FALSE)
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "a"
##
## [[3]]
##      [,1]
## [1,]    0
## [2,]    0
##
## [[4]]
## [1] "b"
##
## [[5]]
## [1] "c"
##
## [[6]]
## [1] 1
##
## [[7]]
## [1] 2
```

Commande `lapply` I

Très puissant !

Applique une fonction à chaque élément d'une liste. La version `sapply` simplifie automatiquement.

```
lapply(maliste,length)

## $numero
## [1] 3
##
## $noms
## [1] 2
##
## $mat
## [1] 4

sapply(list(1,2:4,5:9),function(x) sum(x)/3)

## [1]  0.3333333  3.0000000 11.6666667
```

Plan

Variables et types élémentaires

Vecteurs

Facteurs

Matrices (et tableaux)

Listes

Tableau de données
- Définition
- Manipulation

Plan

Variables et types élémentaires

Vecteurs

Facteurs

Matrices (et tableaux)

Listes

Tableau de données
- Définition
- Manipulation

Tableau de données : définition

Un autre point fort de R

Définition (objet `data.frame`)

C'est une liste à laquelle on impose certaines contraintes afin de rassembler vecteurs et facteurs sous la forme d'un tableau de données.

- Pratiquement, *un tableau de données est une matrice dont les colonnes sont de mode différent,*
- C'est l'objet idéal pour la manipulation de données (forcez-vous à l'utiliser).

Tableau de données : définition

Un autre point fort de R

Définition (objet `data.frame`)

C'est une liste à laquelle on impose certaines contraintes afin de rassembler vecteurs et facteurs sous la forme d'un tableau de données.

- Pratiquement, *un tableau de données est une matrice dont les colonnes sont de mode différent,*
- C'est l'objet idéal pour la manipulation de données (forcez-vous à l'utiliser).

Création de tableau de données

Syntaxe

On peut spécifier le nom des colonnes par le vecteur `row.names` ou directement comme pour une liste :

`data.frame(e1=,e2=,...,row.names=)`

```
age <- c(25,35,32,27,32,40,26,25,26,28,30,NA,36,30,30)
grd <- c("thd","CR","MdC","thd","thd","MdC","MdC","thd","thd","MdC","CR","MdC","CR"
sex <- factor(sample(c(rep("M",3),rep("F",12))))
donnees <- data.frame(age=age,grade=grd,sexe=sex)
head(donnees)
```

```
##   age grade sexe
## 1  25   thd    M
## 2  35    CR    F
## 3  32   MdC    F
## 4  27   thd    F
## 5  32   thd    F
## 6  40   MdC    F
```

Plan

Variables et types élémentaires

Vecteurs

Facteurs

Matrices (et tableaux)

Listes

Tableau de données

- Définition
- Manipulation

Manipulation des éléments du tableau de données I

- Comme une liste !

```
donnees$age
```

```
##  [1] 25 35 32 27 32 40 26 25 26 28 30 NA 36 30 30
```

```
donnees[2]
```

```
##    grade
## 1    thd
## 2     CR
## 3    MdC
## 4    thd
## 5    thd
## 6    MdC
## 7    MdC
## 8    thd
## 9    thd
## 10   MdC
## 11    CR
## 12   MdC
## 13    CR
## 14   thd
```

Manipulation des éléments du tableau de données II

- Mais aussi comme une matrice !

```
donnees[2, ]

##   age grade sexe
## 2  35    CR    F

donnees[1:2, c(1,3)]

##   age sexe
## 1  25    M
## 2  35    F
```

Attention tout de même

Un `data.frame` reste plus proche de la liste...

```
length(donnees)

## [1] 3

dim(donnees)

## [1] 15  3

class(donnees)

## [1] "data.frame"

mode(donnees)

## [1] "list"
```

Le couple `attach/detach` I

les commandes `attach()` / `detach` placent / ôtent les éléments du tableaux de données dans l'itinéraire de recherche

```
attach(donnees,warn.conflicts=FALSE)
grade

##  [1] thd CR  MdC thd thd MdC MdC thd thd MdC CR  MdC CR  thd thd
## Levels: CR MdC thd

age

##  [1] 25 35 32 27 32 40 26 25 26 28 30 NA 36 30 30

detach(donnees)
```

Les fonctions `stack/unstack`

Empile/dépile les colonnes d'une `data.frame`

```
data(cars)
head(cars)
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
## 4     7   22
## 5     8   16
## 6     9   10
```

```
head(stack(cars))
```

```
##   values   ind
## 1      4 speed
## 2      4 speed
## 3      7 speed
## 4      7 speed
## 5      8 speed
## 6      9 speed
```

La fonction `summary`

Réalise un résumé statistique adapté au type de chaque colonne.

```
summary(donnees)
```

```
##       age         grade   sexe
##  Min.   :25.00   CR :3   F:12
##  1st Qu.:26.25   MdC:5   M: 3
##  Median :30.00   thd:7
##  Mean   :30.14
##  3rd Qu.:32.00
##  Max.   :40.00
##  NA's   :1
```

La fonction by

Équivalent de `tapply` pour les tableaux de données ; à utiliser couplé à `with`.

```
attach(donnees,warn.conflicts=FALSE)
by(age,sexe,mean,na.rm=TRUE)

## sexe: F
## [1] 31.18182
## ------------------------------------------------------------
## sexe: M
## [1] 26.33333

detach(donnees)
with(donnees, by(age,grade,mean,na.rm=TRUE))

## grade: CR
## [1] 33.66667
## ------------------------------------------------------------
## grade: MdC
## [1] 31.5
## ------------------------------------------------------------
## grade: thd
## [1] 27.85714
```

La fonction `aggregate`

Permet un `tapply` multivarié ! Attention aux nœuds dans la tête...

```
aggregate(donnees, list(donnees$grade,donnees$sexe), length)

##   Group.1 Group.2 age grade sexe
## 1      CR       F   3     3    3
## 2     MdC       F   4     4    4
## 3     thd       F   5     5    5
## 4     MdC       M   1     1    1
## 5     thd       M   2     2    2
```

Note pour plus tard

Le `data.frame` est l'object idéal pour l'analyse statistique et la manipulation de données

```
anova(lm(age ~ sexe*grade,data=donnees))

## Analysis of Variance Table
##
## Response: age
##             Df  Sum Sq Mean Sq F value Pr(>F)
## sexe         1  55.411  55.411  3.3107 0.1022
## grade        2  56.751  28.376  1.6954 0.2372
## sexe:grade   1   0.919   0.919  0.0549 0.8200
## Residuals    9 150.633  16.737
```