



UNIVERSITÉ
PARIS 7 - DENIS DIDEROT



Introduction à R

Leslie Regad

regad@ebgm.jussieu.fr

Lundi 24 septembre 2007

Qu'est ce que R?

- Environnement statistique et graphique
- Langage interprété pour l'analyse des données
- Version gratuite de S+
- Documentation et packages disponibles:

<http://www.r-project.org>,

Objectif du cours

- ... est simplement de nous familiariser avec R
- Apprendre les bases du langage
- Apprendre à manipuler des données
- Apprendre à faire un graphique

Lancer et quitter R

- Sous linux:
 - Lancer R: **\$R**
 - Quitter R : **q()**
- sauvegarde du workspace
- Langage interprété:
 - ✓ Prompt : **« > »**
 - ✓ Entrer commande et taper **« entrer »**

```
Échier Édition Affichage Terminal Onglets Aide
[regad@balzac ~]$ R

R : Copyright 2005, The R Foundation for Statistical Computing
Version 2.2.0 (2005-10-06 r35749)
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for a HTML browser interface to help.
Type 'q()' to quit R.

> x = "Bienvenue au cours d'intro de R"
> x
[1] "Bienvenue au cours d'intro de R"
> x = 2
> x
[1] 2
> y = x + 2
> y
[1] 4
> m = "maintenant quittons R"
> m
[1] "maintenant quittons R"
> q()
Save workspace image? [y/n/c]: n
[regad@balzac ~]$
```

Les pages d'aides

- Appel de l'aide: `help.start()`, `?fonction`, `help("fonction")`
- Description:
 - ✓ Description: description brève de la fonction
 - ✓ Usage: nom des arguments + valeurs par defaults
 - ✓ Arguments: détail chacun des arguments
 - ✓ Details: description détaillée
 - ✓ Value: le type d'objets retournés
 - ✓ See Also: autres rubriques d'aide proches ou similaires
 - ✓ Examples: des exemples utilisant la fonction

Les Objets en R

- Les différentes classes d'objets :
 - ✓ Les vecteurs, listes, matrices, data frames, facteurs, fonctions
- Les noms des objets :
 - ✓ Ne peuvent pas commencer par un chiffre ou un caractère spécial
 - ✓ Sont sensibles à la case
 - ✓ Certains sont déjà utilisés : q, T, F, D, I, var, mean, ...

L'affectation des variables

- Type des variables : entier, décimal, complexe, logique, caractère
- Assignment: `<-`, `->`, `<<-`, `=`
- Opérations arithmétiques : `+`, `-`, `/`, `*`, `^`
- Fonctions mathématiques usuelles: `log()`, `exp()`, `cos()`, `sin()`

Quelques exemples

```
> x <- 2
```

```
> x
```

```
[1] 2
```

```
> p = 3 * x
```

```
> p
```

```
[1] 6
```

```
> phrase = "Bonjour a tous"
```

```
> print (phrase)
```

```
[1] "Bonjour a tous"
```

```
> help(log)
```

```
> m = log(p)
```

```
> m
```

```
[1] 1.791759
```

Manipulation des types

- **class()** : donne la classe de l'objet

```
> D = matrix(rep(1:2,2),ncol=2)
```

```
> class(D) [1] matrix
```

- **mode()**: donne le type des éléments stockés dans l'objet

```
> mode(D) [1] numeric
```

- **is.type()**: détermination du type

```
> is.numeric(D) [1] True
```

```
> is.character(D) [1] False
```

- **as.type()**: transformation dans un autre type

```
> p = as.character(1:4) [1] "1" "2" "3" "4"
```

Les vecteurs

- Vecteur = liste d'éléments de même type
- Les différents types:
 - ✓ Numeric
 - ✓ Character
 - ✓ Logical
- Valeurs particulières:
 - ✓ **NA** : Valeur manquant (not available)
 - ✓ **NaN** : Pas de nombre (e.g., 0/0)
 - ✓ **-Inf/Inf** : Infini

Création d'un vecteur

- `:` ou `seq()` : génère une série de nombres equidistants

```
> seq(1,5,length = 9)
```

```
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

```
> seq(1,5,by = 0.5)
```

- `numeric()` : création d'un vecteur vide

```
> numeric(5)
```

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

- `c()` : concaténation d'1 ou plusieurs vecteurs

```
> c(seq(1,5), 12:15)
```

```
[1] 1 2 3 4 5 12 13 14 15
```

- `rep()`: réplication d'un vecteur

```
> rep("un",4)
```

```
[1] "un" "un" "un" "un"
```

```
> rep(1:5, each = 2)
```

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

```
> rep(1:5, times=2)
```

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

Les vecteurs

- Définition des étiquettes (=noms) des valeurs du vecteur:

names()

```
> vect1 = (seq(1,5))           [1] 1 2 3 4 5
> names(vect1) = c("val1", "val2", "val3", "val4", "val5")  val1 val2 val3 val4 val5
                                                                    1  2  3  4  5
```

- **length()** : détermine la taille d'un vecteur

```
> length(seq(1,5))           [1] 5
```

- Ajouter une valeur à un vecteur: **append()**

```
> vect =c(1,2, 3)           [1] 1 2 3
>vect = append(vect, c(5,6)) [1] 1 2 3 5 6
```

Calcul sur les vecteurs

- Opérations de base: $+$, $-$, $+$, $*$, $/$

```
> c(1,2,3) + c(1,2,3)      [1] 2 4 6
```

Rmq toujours sur des vecteurs de même taille

- $\text{Sum}()$, $\text{prod}()$, $\text{min}()$, $\text{max}()$, $\text{mean}()$, $\text{sd}()$...

```
> sum(c(1,2,3))           [1] 6
```

```
> min(c(45, 67,32,56))   [1] 32
```

```
> max(c(45, 67,32,56))   [1] 67
```

- $\text{sort}()$, $\text{unique}()$, $\text{intersect}()$, $\text{setdiff}()$, $\text{union}()$, $\text{table}()$...

```
> sort(c(45, 67,32,56))   [1] 32, 45, 56, 67
```

```
> sort(c(45, 67,32,56), decreasing = T) [1] 67, 56, 45, 32
```

Exercices sur les vecteurs

- Créer le vecteur d'entiers x allant de 1 à 20
- Créer le vecteur Sx contenant le sinus de chaque élément de x
- Calculer la somme et la somme cumulée de Sx .
(*sum()*, *cumsum()*)
- Calculer la moyenne, la variance et la valeur maximale de Sx

Matrices et tableaux

- Matrices = Vecteurs avec des dimensions. Elles contiennent des numerics, characters ou logicals

- **matrix()** : création d'une matrice

> M = matrix(c(1,2,3,4,5,6), nrow = 2, ncol =3)

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

- **diag()** : récupération de la diagonale de la matrice

- **colnames()/ rownames()** : noms des lignes et colonnes

> colnames(M) = c("Col1", "Col2", "Col3")

> rownames(M) = c("Row1", "Row2")

```
      Col1 Col2 Col3
Row1    1   3   5
Row2    2   4   6
```

Matrices et tableaux

- **cbind()** : concaténation de plusieurs vecteurs (en colonnes)

> M = cbind(c(1,2),c(3,4),c(5,6))

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

- **rbind()** : concaténation de plusieurs vecteurs (en lignes)

> Mat = rbind(c(1,2),c(3,4),c(5,6))

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

- **dim()** : dimension des matrices

> dim(Mat) [1] 3 2

1^{er} élément = nombre de lignes

2nd élément = nombre de colonnes

Calculs matriciels

- `%*%` : Multiplication de 2 matrices
- `t()` : Transposée d'une matrice
- `eigen()` : Valeurs et vecteurs propres
- `det()` : Déterminant d'une matrice
- `solve()` : Inversion et résolution d'un système linéaire

```
> M = cbind(c(1,2),c(,3,4),c(5,6))
```

```
> M %*% t(M)
```

```
> M = cbind(c(1,-2,0,0),c(-2,1,-2,0),c(0,-2,1,-2),c(0,0,-2,1))
```

```
> b = rep(1,4)
```

```
> solve(A,b)
```

Exercices sur les matrices

- Créer la matrice de taille 10x10 avec des 1 sur la diagonale et des 0 ailleurs
- Créer une matrice A de taille 10*10 dont les éléments suivent une loi normale de moyenne nulle et de variance 5

Extraction de n valeurs d'une loi normale de moyenne m et de variance v :

`rnorm(n, m, sqrt(v))`

- Calculer le déterminant de A

Les data frames

- Data frames = matrices dont les colonnes peuvent être de différents types.

- `data.frame()` : création d'un data frame

```
> D = data.frame(noms = c("toto","titi","tutu"), ages = c(8,9,10))
```

```
> D
```

	[1] noms	ages
1	toto	8
2	titi	9
3	tutu	10

- `cbind()`, `rbind()`

- `dim()`

Indexation

- Indexation = extraction d'un ou plusieurs éléments d'un vecteurs ou matrices
- Plusieurs types d'indexation en utilisant l'opérateur []
 - ✓ Par un vecteur d'entier positif
 - ✓ Par un vecteur d'entier négatif
 - ✓ Par un vecteur logique
 - ✓ Par un vecteur de nom

L'indexation des vecteurs

- L'indexation par un vecteur positif

```
> Vect1 = seq(1,5)           [1] 1 2 3 4 5
> Vect2 = Vect1[1:4]        [1] 1 2 3 4
```

- L'indexation par un vecteur négatif : permet l'exclure 1 ou plusieurs valeurs

```
> Vect3 = Vect1[-c(2,4)]
[1] 1 3 5 ##exclusion des éléments 2 et 4
```

L'indexation des vecteurs

- L'indexation par un vecteur logique:

which(): donne les positions vérifiant la condition

```
> Vect1 > 3
```

```
[1] False False False True True
```

```
> Vect4 = Vect1[which(Vect1 > 3)]
```

```
[1] 4 5 ### donne les éléments de Vect1 qui  
vérifie la condition précisée (ici > 3)
```

- L'indexation par un vecteur nom (utilité de la fonction `names()`)

```
> name(Vect1) = paste("Ele", 1:5, sep = "")
```

```
Ele1 Ele2 Ele3 Ele4 Ele5  
1 2 3 4 5
```

```
> Vect2 = Vect1[c("Ele1", "Ele3")]
```

```
[1] 1 3
```

Exercices sur l'indexation des vecteurs

- A partir du vecteur Sx :
- Créer le vecteur $xneg$ des indices des valeurs négatives de Sx (*which()*)
- Remplacer les valeurs de Sx supérieur à sa moyenne par 40. Nommer ce vecteur SSx

L'indexation des matrices et data frames

- L'indexation similaire à celle des vecteurs. Elle se fait pas l'intermédiaire de l'opérateur `[]`, mais nécessite 2 indices (lignes et colonnes)

```
> Mat = cbind(c(1,2,3), c(4,6,7),c(4,8,5), c(2, 5, 7))
```

```
> colnames(Mat) = c("col1", "col2", "col3", "col4")
```

```
> rownames(Mat) = c("ligne1", "ligne2", "ligne3", )
```

```
> Mat2 = Mat1[1:2, 1:2]      ###extraction des 2 premières lignes et colonnes
```

```
> Mat2 = Mat1[c("ligne1", "ligne2"), ]      ###extraction des 2 premières lignes
```

Exercices sur l'indexation des matrices

- Calculer le nombre de nombre positif et négatif de la matrice A.
- Quelles sont les position des éléments négatifs de A.
which() et which(, arr.ind=T)
- Récupérer la sous-matrice As:
 - 5 premières lignes
 - 5 dernières colonnes
 - Remplacer les nombres négatifs de As par 0

Les facteurs

- Les facteurs = représentation des variables qualitatives.

Ex: la couleur des yeux, le niveau de douleur...

```
> douleur = c(0, 3, 2, 2, 1)
```

```
> fdouleur = factor (douleur)
```

```
[1] 0 3 2 2 1
```

```
Levels: 0 1 2 3
```

```
> levels(fdouleur) = c("rien", "leger", "moyen", "fort")
```

```
> fdouleur
```

```
[1] rien fort moyen moyen leger
```

```
Levels: rien leger moyen fort
```

Les listes

- Listes = vecteurs dont les éléments ne sont pas nécessairement de même type.

- **list()** : création d'une liste

```
> liste1 = list(Element1 = c("A1","A2","A3"),Element2 = c("B1","B2"))
```

- Extraction d'un élément se fait généralement par son **nom** (opérateur **\$**) ou par sa **position** ou **[[]]**

```
> liste1$Element1   ou liste1[1]   ou liste1[[1]]           [1] "A1" "A2" "A3"
```

```
> liste1$Element2   ou liste1[2]   ou liste1[[2]]           [1] "B1" "B2"
```

- Extraction d'un sous-élément:

```
> liste1$Element1[1]           [1] "A1"
```

Comparaison

Inférieur	<	Inférieur ou égal	<=
Supérieur	>	Supérieur ou égal	>=
égal	==	différent	!=

- Opérateurs qui travaillent sur les vecteurs par éléments

> Vect1 = 1:5

[1] 1 2 3 4 5

> Vect1 >= 2

[1] False True True True True

Opérateurs logiques

ET	&	&&
OU		

- Opérateurs qui travaillent sur les vecteurs par éléments

```
> Vect1 = 1:5           [1] 1 2 3 4 5
> Vect1 >= 2 & Vect1 <5 [1] False True True True False
```

- Pour les tests: **&&**, **||**

```
> Vect1 = 1:5           [1] 1 2 3 4 5
> x = -1
> (x>0) & (log(x) > 0.5) [1] FALSE
Warning message: NaNs produced in: log(x)
> (x>0) && (log(x) > 0.5) [1] FALSE
```

Boucle for

- `for (cond) { expr }`

```
> comp = 0
```

```
> for (i in seq(5,8)){
```

```
+   comp = comp + i
```

```
+   print (comp) }
```

```
[1] 5
```

```
[1] 11
```

```
[1] 18
```

```
[1] 26
```

Boucle while

- `while (cond) { expr }`

```
> comp = 0
```

```
> while (comp < 10){
```

```
+   comp = comp + 2
```

```
+   print (comp) }
```

```
[1] 2
```

```
[1] 4
```

```
[1] 6
```

```
[1] 8
```

Test if

- `if (cond) { expr}`

```
> Vect = c("fort", "fort", "leger", "leger", "moyen", "leger", "rien")
```

```
> compt = 0
```

```
> for (l in Vect){
```

```
+   if (l != "fort") {
```

```
+     compt = compt + 1}
```

```
> compt
```

```
[1] 5
```

Exercices sur les boucles

- Calculer la valeur moyenne de chaque colonne de la matrice A.

Les boucles implicites

- **Avantage:** Application d'une fonction à tous les éléments d'un vecteur ou d'une liste.
- **lapply(X,FUN,...):** renvoie une liste de même taille que X dont chaque élément est le resultat de l'application de la fonction FUN à l'élément correspondant de X
- **apply(X,MARG,FUN,...):** renvoie un vecteur ou autres de valeurs obtenues en appliquant la fonction FUN aux marginales MARG de la matrice X

```
> apply(matrix(c(1,2,1,2),ncol = 2), 1,sum) ###somme des lignes de la matrice  
[1] 2 4
```

Exercices sur les boucles

- Calculer la valeur moyenne de chaque colonne de la matrice A.

Les fonctions

- Eviter la redondance dans les programmes
- Améliorer la visibilité du programme
- Découper le programme en liste d'action

- Syntaxe:

```
Function_name = function(arg1, arg2){  
    expr,  
    return(expr_return)  
}
```

- Appel de la fonction:

```
Val1 = Function_name(arg1, arg2)
```

Les fonctions

```
> my_function = function( x , y ) {  
+   z = y / x * 100  
+   return (z)  
+ }  
  
> val1 = my_function(12,50)  
  
[1] 24
```

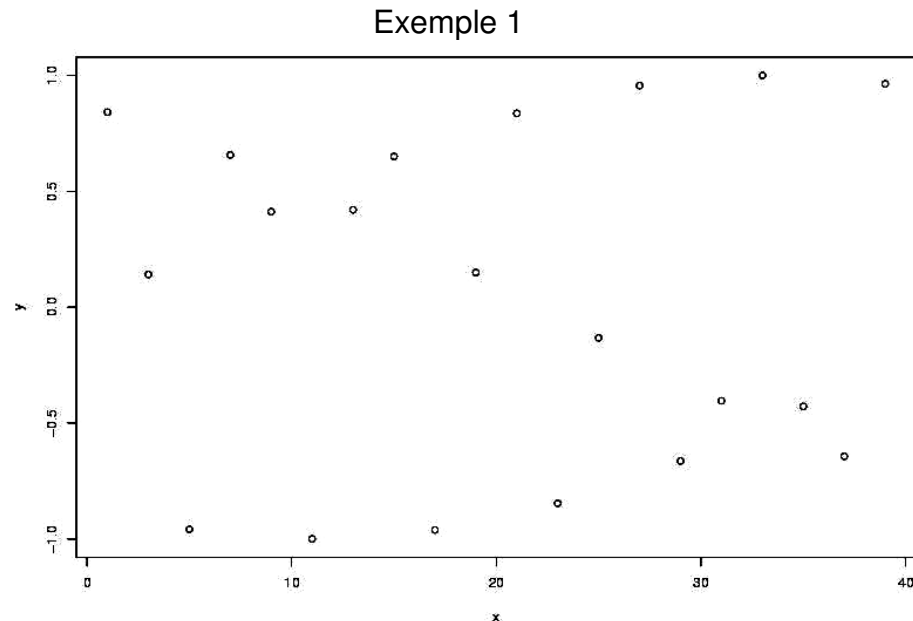
```
> my_function2 = function (x , y) {  
+   z = y / x * 100  
+   Mat = cbind (y , x , z)  
+   L = apply ( Mat, 2, mean)  
+   return(L)  
+ }
```

###ici x et y sont des vecteurs

Les graphiques: la fonction plot

- La fonction générique : `plot (x,y, ylim = num, xlim = num, ylab = char, xlab = char, main = char...)`
- Enregistrer : `postscript("plot.eps"); plot() ; dev.off()`
`jpg ("plot.jpg"); plot(); dev.off()`

```
> x = seq(1,40,by = 2)
> y = sin(x)
> postscript("graph1.eps")
> plot(x,y, main = "exemple 1",
ylab = "y", xlab = "x" )
> dev.off()
```

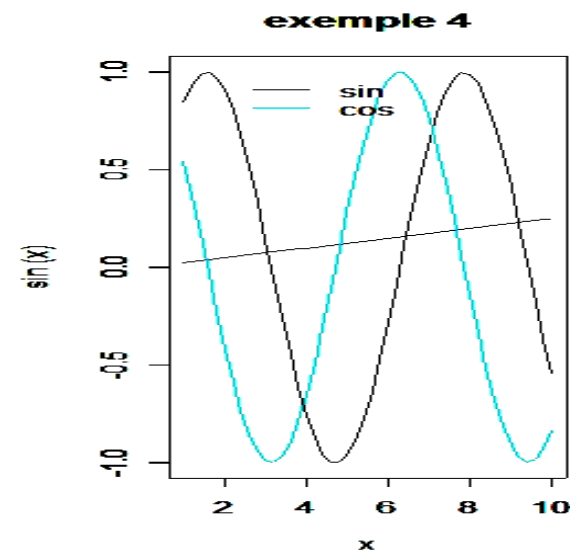
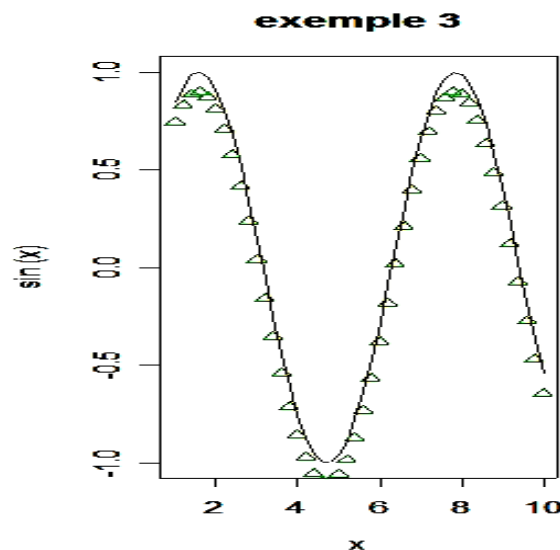


Les graphiques: la fonction plot

- plus d'info:
 - ✓ `Points()`, `lines()`
 - ✓ `Col`, `type`, `legend`, `axis`, `text`, `pch`, ...
 - ✓ `Par`
 - `Par(new=T)`, `par(mfrow=c(,))`,
 - `Cex`,...

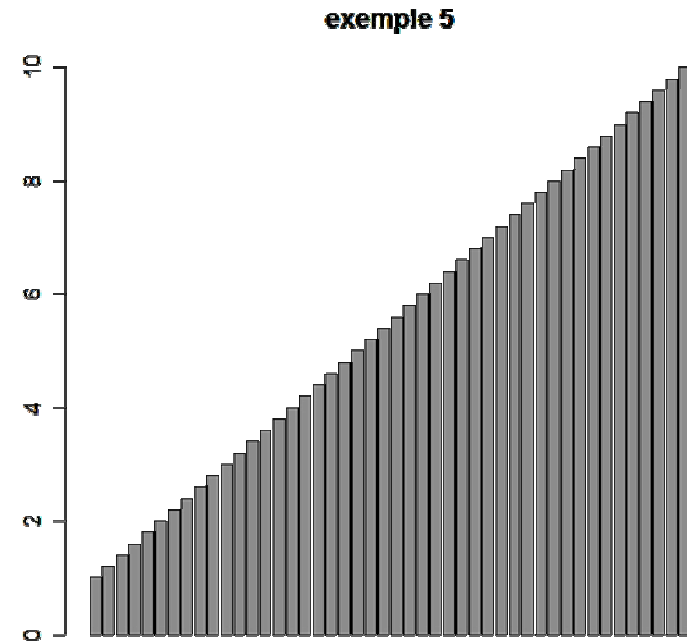
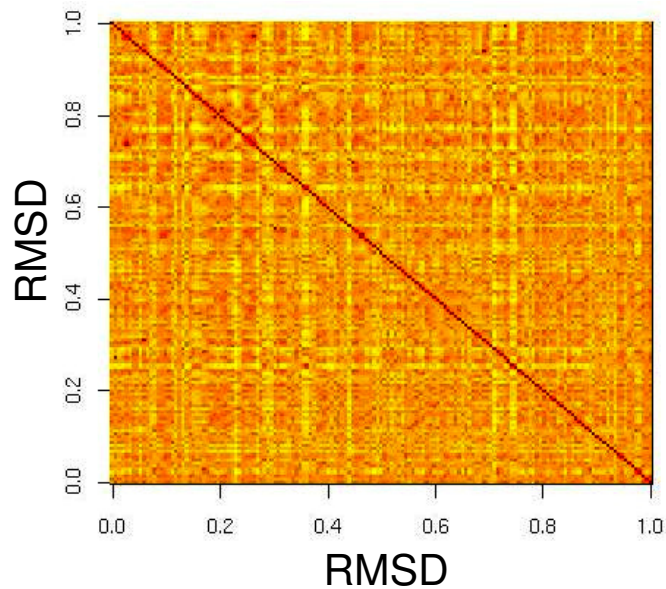
Les graphiques

- > par(mfrow = c(1,2))
- > plot(x,sin(x), main= "exemple 3", ylab= "sin (x)", xlab = "x", col = "red", type = "l")
- > points(x,(sin(x)-0.1),col=3,pch = 24)
- > plot(x,sin(x), main= "exemple 4", ylab= "sin (x)", xlab = "x", col = "red", type = "l")
- > lines(x, x/40,col=4)
- > par(new=T)
- > plot(x,cos(x), main= "exemple 4", ylab="sin (x)", xlab = "x", col = 5, type = "l")
- > legend(2,1,col=c(2,5),c("sin","cos"), lty=1, bty="n")



Les graphiques: la fonction plot

- autres types de plots:
 - ✓ Histogramme : `hist()`
 - ✓ Horizontal bars plot : `barplot()`
 - ✓ `Surface()`
 - ✓ Image de matrice: `image()`



```
> x = seq(1,10,by = 0.2)  
> barplot(x, main = "exemple 5")
```

```
> image( t(Mat) [,nrow(Mat):1], ylab="RMSD",  
+ xlab="RMSD")
```

Exportation des données

- **cat()** : écriture générique dans un fichier
- **write()** : écriture d'un vecteur/matrice dans un fichier
- **write.table()** : écriture d'un data frame dans un fichier

```
> x = 1:10
```

```
> write(x, file = "test.dat")
```

```
> x = matrix( 1:10,ncol = 5)
```

```
> write( t(x), file = "Mat.dat")
```

```
> D = data.frame ( noms = c("toto","tata",  
+ "titi"), age = c(3,4,5))
```

```
> write.table(D, file = "Data.dat")
```

Importation des données

- `scan()` : lecture de données à partir d'un fichier
- `scan` : importation d'un vecteur
- `read.table()` : importation d'un data frame/ matrice
- `readLines()` : lire un fichier texte ligne par ligne

```
> scan("test.dat")
```

```
> x = matrix( 1:10,ncol = 5)
```

```
> read.table ("Mat.dat")
```

```
> read.table(D, file = "Data.dat",header=T)
```