

# Initiation au langage R.

CIRM : Rencontre Algorithmique et Programmation

Thierry Dumont

Modélisation aléatoire de Paris Ouest Nanterre La Défense (MODAL'X)

April 21, 2014

## 1 Introduction

## 2 Formalisme

- Les conteneurs
- Opérateurs logiques, boucles et conditions
- Les fonctions

## 3 Probabilités

## 4 L'aide en ligne

## 5 Les librairies

## 6 Enregistrer/charger des données

Vous pouvez récupérer :

- cette présentation,
- l'énoncé de TP d'introduction
- les codes R associés

Dans la partie "enseignements" de :

*[https : //sites.google.com/site/thierrydumontmaths/](https://sites.google.com/site/thierrydumontmaths/)*

# Présentation

R est ...

un logiciel libre de traitement des données et d'analyse statistiques via un langage de programmation "interprété" (ne nécessite pas de compilation).

# Présentation

## R est ...

un logiciel libre de traitement des données et d'analyse statistiques via un langage de programmation "interprété" (ne nécessite pas de compilation).

## Mais c'est surtout ...

- La plus large collection d'outils statistiques, en perpétuelle évolution
- Un langage simple centré sur la notion de vecteurs, sans typage ou déclaration de variables,
- Une aide en ligne simple souvent illustrée par de nombreux exemples.

# Principales Fonctionnalités

- Manipulation de données (stockage, outils statistiques)
- Algèbre linéaire
- Large variété d'outils graphiques
- Un très grand nombre de bibliothèques (Open Source (Licence GNU/GPL))
- Possible interface avec *C*, *C++*, *Fortran*, *Java*, ...

# Inconvénients

## Les inconvénients

- R est un peu lent (Boucles, calcul matriciel),
- l'espace mémoire alloué à R est limité.
- On ne peut sauvegarder le code R exécuté : Editer et Sauvegarder le script dans un éditeur de texte (Notepad ++, Tinn-R).

# Présentation

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

Point de départ pour télécharger gratuitement R sur les principales plateformes : Linux, Windows, MacOS...

<http://cran.r-project.org/>

Libraries Documentation



```

oooooooooooooooo
ooo
ooo
ooo

```

## Le logiciel

R se présente comme un invité de commande. Cependant des interfaces graphiques sont disponibles pour

- Windows *Rgui* : Interface graphique basique facilitant le téléchargement des librairies
- MacOS *R.app*: Plus élaborée.

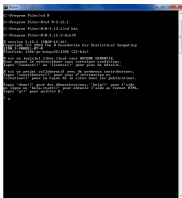


Figure: Invité de commandes

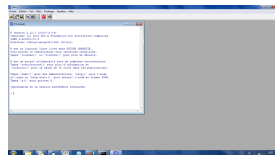


Figure: RGui sur Windows

oooooooooooooooo

ooo

ooo

# Exemples

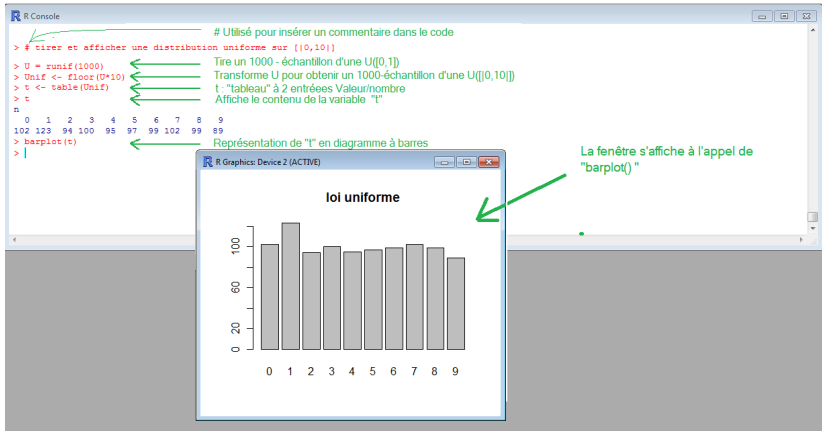
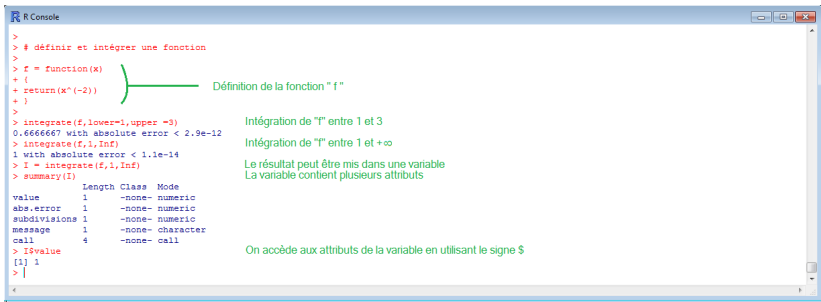


Figure: Tirer et représenter une loi uniforme sur  $\{0, \dots, 10\}$

```
oooooooooooooooo  
ooo  
ooo
```

# Exemples



```
>  
> # définir et intégrer une fonction  
>  
> f = function(x)  
+ {  
+   return(x^(-2))  
+ }  
>  
> integrate(f,lower=1,upper =3)  
0.6666667 with absolute error < 2.9e-12  
> integrate(f,1,Inf)  
1 with absolute error < 1.1e-14  
> I = integrate(f,1,Inf)  
> summary(I)  
      Length Class  Mode  
value      1      -none- numeric  
abs.error   1      -none- numeric  
subdivisions 1      -none- numeric  
message     1      -none- character  
call        4      -none- call  
> I$value  
[1] 1  
> |
```

Définition de la fonction "f"

Intégration de "f" entre 1 et 3

Intégration de "f" entre 1 et +∞

Le résultat peut être mis dans une variable  
La variable contient plusieurs attributs

On accède aux attributs de la variable en utilisant le signe \$

Figure: Définir et intégrer une fonction

```

oooooooooooooooo
ooo
ooo
ooo

```

# Exemples

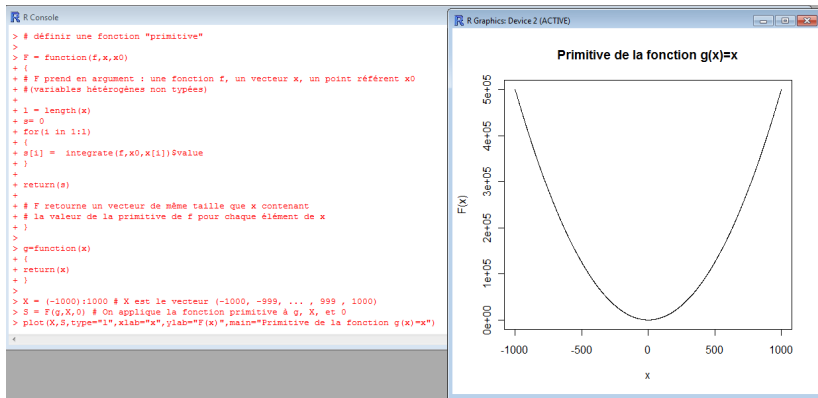
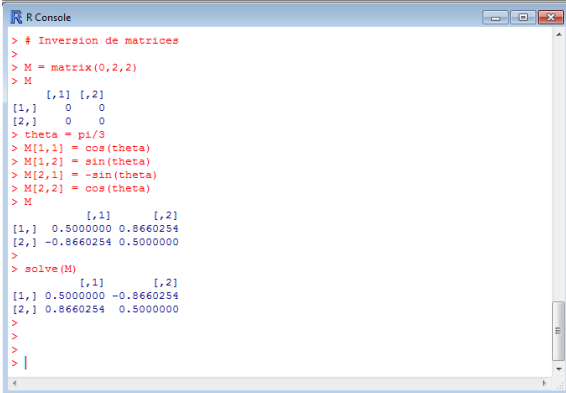


Figure: La fonction primitive

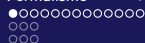
```
oooooooooooooooo  
ooo  
ooo  
ooo
```

# Exemples



```
R Console  
  
> # Inversion de matrices  
>  
> M = matrix(0,2,2)  
> M  
      [,1] [,2]  
[1,]    0    0  
[2,]    0    0  
> theta = pi/3  
> M[1,1] = cos(theta)  
> M[1,2] = sin(theta)  
> M[2,1] = -sin(theta)  
> M[2,2] = cos(theta)  
> M  
      [,1] [,2]  
[1,] 0.5000000 0.8660254  
[2,] -0.8660254 0.5000000  
>  
> solve(M)  
      [,1] [,2]  
[1,] 0.5000000 -0.8660254  
[2,] 0.8660254 0.5000000  
>  
>  
>  
> |
```

Figure: L'inversion de matrices



## 1 Introduction

## 2 Formalisme

### ■ Les conteneurs

■ Opérateurs logiques, boucles et conditions

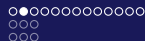
■ Les fonctions

## 3 Probabilités

## 4 L'aide en ligne

## 5 Les librairies

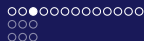
## 6 Enregistrer/charger des données



# Les vecteurs

## Déclaration d'un vecteur

- `#` Générer un vecteur de nombres pseudo-aléatoires issus  
d'une loi normale centrée réduite.  
`x = rnorm(50)`
- `#` Un nombre ou une chaîne de caractères simple:  
`x = 10` `#` est un vecteur de taille 1  
`x = "R"` `#` est un vecteur de taille 1
- `#` Définir un vecteur nul de taille donnée  
`x = rep(0, 20)` `#` est le vecteur nul de taille 20  
`x = numeric(20)` `#`idem  
`x = 1 : 20` `#` est le vecteur  $(1, 2, \dots, 20)$   
`seq(from = -5, to = 10)` `#` est le vecteur  $(-5, -4, \dots, 10)$

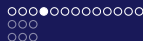


# Les vecteurs

Éléments et attributs :  $x = seq(from = -5, to = 5)$

- #Accéder à un élément (le 1er index est "1") :  
 $x[5]$  # Retourne  $-1$
- #Accéder à une liste d'éléments :  
 $x[1 : 5]$  # Retourne le vecteur  $(-5, -4, -3, -2, -1)$
- #Longueur d'un vecteur :  
 $length(x)$  # Retourne 11
- #Element maximal :  
 $max(x)$  # Retourne 5  
 $which.max(x)$  # Retourne 11
- #Fonction "which" :  
 $which(x > 1)$  # Retourne indexes des éléments de  $x$  sup. à 1.

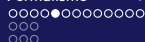




# Les vecteurs

Éléments et attributs :  $x = \text{seq}(\text{from} = -5, \text{to} = 5)$

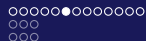
- #Trier les éléments :  $y = 5 : 1$   
`sort(y)` #Retourne le vecteur trié (1, 2, 3, 4, 5)
- # somme d'un vecteur :  
`sum(x)` #Retourne 0
- # somme cumulée d'un vecteur :  
`cumsum(x)` #Retourne le vecteur  $(-5, -9, -12, \dots, 0)$
- # produit d'un vecteur :  
`prod(x)` #Retourne 0



# Les vecteurs

## Opérations sur les vecteurs ( $x = (-5) : 5$ )

- `#Multiplication par un scalaire :`  
 $x * 2$  `#` retourne le vecteur  $(-10, -8, \dots, 8, 10)$
- `#Multiplication de deux vecteurs de même longueur :`  
 $y = 1 : 11$   
 $x * y$  `#` retourne le vecteur  $(-5, -8, -9 \dots, 50, 55)$   
 $x + y$  `#` retourne le vecteur  $(-4, -2, \dots, 14, 16)$
- `#Concaténation :`  
 $c(x, y)$  `#` met les vecteurs  $x$  et  $y$  bout à bout.
- `#Réaffectation des coordonnées :`  
 $x[1 : 3] = c(-1, 3, 2)$  `#` Les 3 premières coordonnées de  $x$  sont maintenant  $(-1, 3, 2)$   
 $x[1 : 3] = 1$  `#` Passe à 1 les 3 premières coordonnées de  $x$

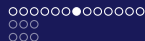


# Les matrices

Une matrice n'est autre qu'un vecteur dont chaque élément est associé à une coordonnée.

Déclaration:

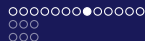
- $M = \text{matrix}(0, nrow = 3, ncol = 3)$
- $M = \text{matrix}(c(3, 4, 5, 6), 2, 2, byrow = TRUE)$



# Les matrices

Éléments et attributs :  $M = \text{matrix}(c(3, 4, 5, 6, 7, 8), 3, 2, \text{byrow} = \text{TRUE})$

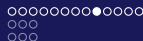
- #Accéder à un élément de la matrice  
 $M[2, 1]$  #Retourne 5
- #Accéder à une ligne colonne  
 $M[, 1]$  #Retourne le **vecteur** (3,5,7)  
 $M[2, ]$  #Retourne le **vecteur** (5,6)
- #Accéder à certaines lignes/colonnes de la matrice  
 $M[1 : 2, ]$  #Retourne une sous matrice 2 lignes 2 colonnes
- #Insérer des conditions  
 $M[M[, 1] < 6, ]$  #Retourne les lignes correspondant à  
"  $\text{which}(M[, 1] < 6)$  "



# Les matrices

Éléments et attributs :  $M = \text{matrix}(c(3, 4, 5, 6, 7, 8), 3, 2, \text{byrow} = \text{TRUE})$

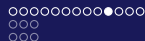
- #Dimensions d'une matrice  
 $\text{dim}(M)$  #Retourne le vecteur (3,2)
- #nombre de lignes  
 $\text{nrow}(M)$  #Retourne 3
- #nombre de colonnes  
 $\text{ncol}(M)$  #Retourne 2



# Les matrices

Opérations élémentaires.  $M = \text{matrix}(c(2, 3, 4, 5), 2, 2, \text{byrow} = \text{TRUE})$

- #Multiplication par un scalaire :  
 $M * 2$  # retourne la matrices dont les coeffs. sont multipliés par 2
- #Multiplication/addition de deux matrices de même dimensions :  
 $M2 = \text{matrix}(1, 2, 2)$   
 $M * M2$  # Multiplie les coefficients termes à termes.  
 $M + M2$  # Additionne les coefficients termes à termes.
- #Concaténation :  
 $\text{rbind}(M, M2)$  # crée la matrice  $\begin{pmatrix} M \\ M2 \end{pmatrix}$   
 $\text{cbind}(M, M2)$  # crée la matrice  $(M | M2)$



# Les matrices

Opérations matricielles.  $M = \text{matrix}(c(2, 3, 4, 5), 2, 2, \text{byrow} = \text{TRUE})$

- #Transposée :  $t(M)$
- #Produit matriciel
  - $M2 = \text{matrix}(1, 2, 3);$

$M \% * \% M2$

- $u = c(1, 1);$

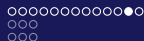
$M \% * \% u$

- # Déterminant :  $\text{det}(M)$
- # Inverse :
  - # de matrice :  $\text{solve}(M)$  ou  $\text{ginv}(M)$  #(librairie MASS)
  - # de système :  $u = c(1, 1), \text{solve}(M, u).$

# Les listes

- Les listes peuvent contenir des données hétérogènes.
- Utilisées notamment dans les fonctions





# Les listes

## Déclaration

- # Liste vide :  $L = \{\}$
- # Liste de 3 éléments  $x$ : vecteur,  $M$ : matrice, *word*: Chaîne de caractères  
 $L = \{x, M, word\}$
- # Nommer les éléments de la liste  
 $L = \{vect = x, mat = M, text = word\}$
- # Accéder aux éléments  
 $L[[2]]$  # Retourne  $M$
- # Lorsque les éléments ont été nommés  
 $L$text$  # Retourne *word*
- # Rajouter/Modifier un élément  
 $L[[4]] = M2$



# Autres Conteneurs

D'autres conteneurs existent

Parmi lesquels

- Les "array" : extension des matrices (matrices à  $D$  coordonnées)
- Les "data.frame" : défini des tableaux possiblement hétérogènes
- Et beaucoup d'autres...



## 1 Introduction

## 2 Formalisme

### ■ Les conteneurs

### ■ Opérateurs logiques, boucles et conditions

### ■ Les fonctions

## 3 Probabilités

## 4 L'aide en ligne

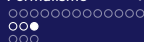
## 5 Les librairies

## 6 Enregistrer/charger des données



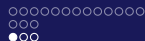
# Opérateurs logiques

opérateur	Signification
<code>==, &gt;, &lt;, &lt;=, &gt;=</code>	Opérateurs de comparaison
<code>!=</code>	différent de
<code> ,   </code>	OU
<code>&amp;, &amp;&amp;</code>	ET
<code>!</code>	négation



# Boucles et conditions

- *for*(compteur in vecteur){ #instructions }  
# Exemple : *for*(i in 1 : 10){ *print*(i) }
- *while*(condition){ #instructions }  
# Exemple : *i* = 1; *while*(*i* <= 10){ *print*(*i*); *i* ++ }
- *if*(condition){ #instructions<sub>1</sub> }else{ #instructions<sub>2</sub> }  
# Exemple : *x* = *runif*(30); *if*(*sum*(*x*) < 10){ *hist*(*x*) }



## 1 Introduction

## 2 Formalisme

- Les conteneurs
- Opérateurs logiques, boucles et conditions
- Les fonctions

## 3 Probabilités

## 4 L'aide en ligne

## 5 Les librairies

## 6 Enregistrer/charger des données



# Les fonctions

## Syntaxe

```
f = function(v=rep(1,2),M=matrix(1:4,2,2),affiche=TRUE)
{
  # Variables : v,M,affiche
  if(affiche)
  {
    print(v)
    print(M)
  }
  if(!is.matrix(M))
  {
    cat("M n'est pas une matrice")
    return(0)
  }
  if(nrow(M) != ncol(M))
  {
    cat("Erreur la matrice n'est pas carré")
    return(1)
  }
  if(det(M) == 0)
  {
    cat("Erreur matrice non inversible")
    return(2)
  }
  if(length(v) != nrow(M)[1])
  {
    cat("problème de dimension")
    return(3)
  }
  return(solve(M,v))
}
```

Valeurs par défaut

Sortie de la fonction

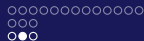


# Les fonctions

## Syntaxe

```
> X=f ()  
[1] 1 1  
      [,1] [,2]  
[1,]    1    3  
[2,]    2    4  
> X  
[1] -0.5  0.5
```





# Les fonctions

## Syntaxe

```
> M=matrix(0,2,2)
> M[1,1]=1
> M[1,2]=-1
> M[2,1]=-1
> M[2,2]=2
> v=c(1,-1)
> X=f(v,M)
[1] 1 -1
      [,1] [,2]
[1,] 1 -1
[2,] -1 2
> X
[1] 1 0
> |
```



# Les fonctions

## Syntaxe

```
> M=matrix(0,2,2)
> M[1,1]=1
> M[1,2]=-1
> M[2,1]=-1
> M[2,2]=2
> v=c(1,-1)
> X=f(v=v,M=M,affiche=FALSE)
> X
[1] 1 0
```



# Les fonctions

## Syntaxe

```
> M=matrix(0,2,2)
> v=c(1,-1)
> X=f(v=v,M=M,affiche=FALSE)
Erreur matrice non inversible>
```



# Les fonctions

## Syntaxe : sorties multiples

```
Analyse=function(M)
{
  if(!is.matrix(M)){ cat("M n'est pas une matrice"); return(0);}

  dimM = dim(M)
  TrM = sum(diag(M))
  detM = det(M)
  if(detM!=0) invM=solve(M)
  else invM=NULL
  S=list(mat=M,nRows=dimM[1],nCols=dimM[2],Trace = TrM, det=detM, inverse=invM)
  return(S)
}
```



# Les fonctions

## Syntaxe : sorties multiples

```
> M=matrix(1:4,2,2)
> A=Analyse(M)
> A$mat
      [,1] [,2]
[1,]     1     3
[2,]     2     4
> A$nRows
[1] 2
> A$nCols
[1] 2
> A$Trace
[1] 5
> A$det
[1] -2
> A$inverse
      [,1] [,2]
[1,]    -2  1.5
[2,]     1 -0.5
```

## 1 Introduction

## 2 Formalisme

- Les conteneurs
- Opérateurs logiques, boucles et conditions
- Les fonctions

## 3 Probabilités

## 4 L'aide en ligne

## 5 Les librairies

## 6 Enregistrer/charger des données

# Variables aléatoires

loi	nom	Paramètres	valeurs par défaut
Beta	beta	shape1,shape2	
Binomiale	binom	size,prob	
Cauchy	cauchy	location,scale	0, 1
Exponentielle	exp	rate	1
Fisher	f	df1,df2	
Gamma	gamma	shape,rate	·, 1
Géométrique	geom	prob	
HyperGéom.	hyper	m,n,k	
$\chi^2$	chisq	df	
Normale	norm	mean,sd	0, 1
Poisson	pois	lambda	
Uniforme	unif	min,max	0, 1



# Variables aléatoires

R permet de simuler des variables aléatoires et d'accéder à certaines de leurs caractéristiques. Pour chaque distribution la syntaxe est la suivante:

- *dloi* : #fonction de densité. (Variable discrète : retourne  $P(X = \cdot)$ ).
- *ploi* : #fonction de répartition. ( retourne  $P(X \leq \cdot)$ ).
- *qloi* : #fonction quantile
- *rloi* : #tire des réalisations aléatoires indépendantes de la distribution "loi"

La fonction *sample*(*vec*, *n*, *prob*) permet de tirer aléatoirement parmi les éléments de *vec*



oooooooooooo

ooo

ooo

# Variables aléatoires

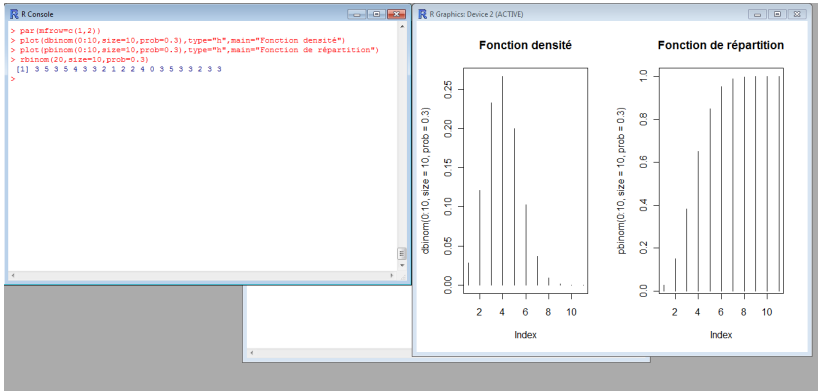


Figure: Loi binomiale

# Statistiques Descriptives

Si  $X$  est un vecteur de données:

- `ecdf(X)` : #Retourne la fonction de répartition empirique de  $X$
- `mean(X)` : #Retourne la moyenne de  $X$
- `var(X)` : #Retourne la variance de  $X$
- `hist(X)` : #Affiche l'histogramme de  $X$
- ...

## 1 Introduction

## 2 Formalisme

- Les conteneurs
- Opérateurs logiques, boucles et conditions
- Les fonctions

## 3 Probabilités

## 4 L'aide en ligne

## 5 Les librairies

## 6 Enregistrer/charger des données



# L'aide en ligne

De l'invité de commande R on peut (on doit) accéder aux informations relatives à chaque fonction utilisée via la commande :

*> ?NomDeLaFonction*

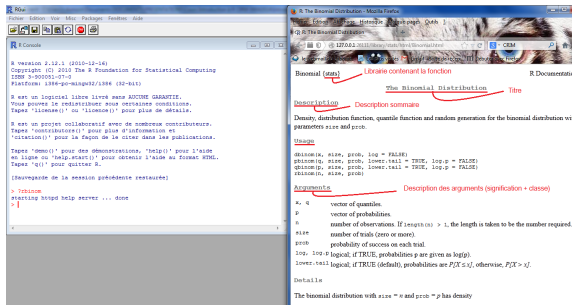


Figure: Description Aide R



# L'aide en ligne

De l'invité de commande R on peut (on doit) accéder aux informations relatives à chaque fonction utilisée via la commande :

`> ?NomDeLaFonction`

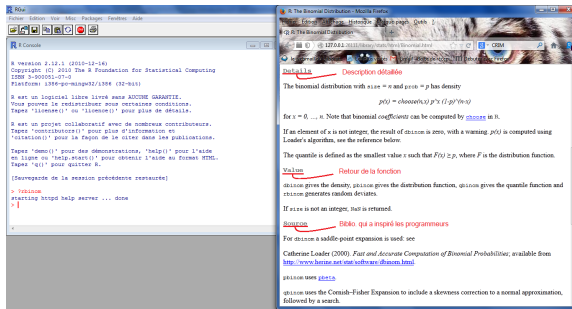


Figure: Description Aide R



# L'aide en ligne

De l'invité de commande R on peut (on doit) accéder aux informations relatives à chaque fonction utilisée via la commande :

**>?NomDeLaFonction**

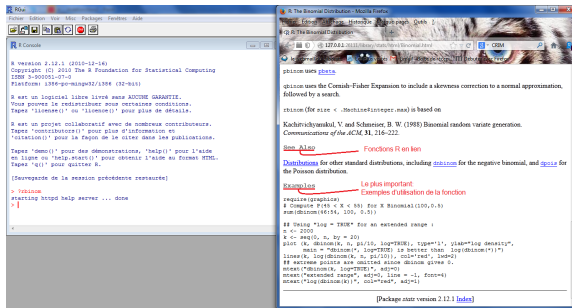


Figure: Description Aide R



## L'aide en ligne

Lorsqu'on cherche une fonction particulière : utilisation de la commande :

>?? "TermesDeRecherche"

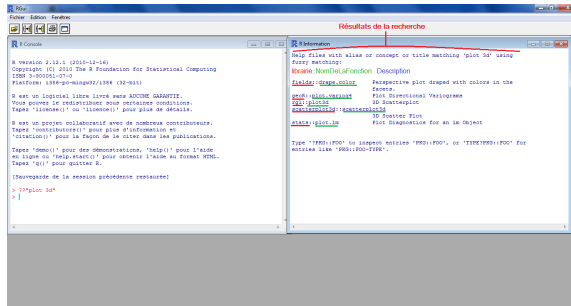


Figure: Recherche fonction

ooooooooooooo  
ooo  
ooo  
ooo

## 1 Introduction

## 2 Formalisme

- Les conteneurs
- Opérateurs logiques, boucles et conditions
- Les fonctions

## 3 Probabilités

## 4 L'aide en ligne

## 5 Les bibliothèques

## 6 Enregistrer/charger des données





# Installer et charger une librairie

## Installer une librairie

La commande pour installer une librairie est la suivante:  
*install.packages(" NomDeLaLibrairie")*

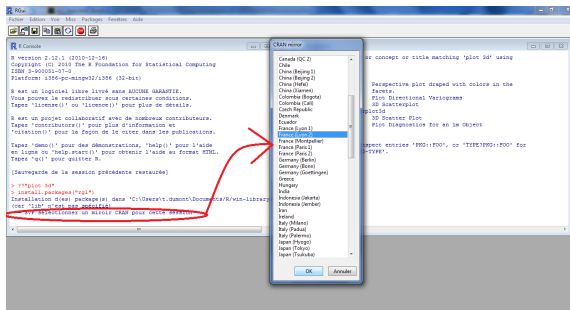


Figure: Installation librairie



# Installer et charger une librairie

## Installer une librairie

La commande pour installer une librairie est la suivante:  
*install.packages("NomDeLaLibrairie")*

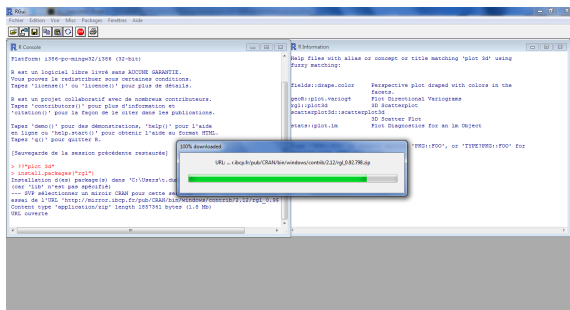


Figure: Installation librairie

# Installer et charger une librairie

## Installer une librairie

La commande pour installer une librairie est la suivante:  
*install.packages("NomDeLaLibrairie")*

```

R
Fichier Édition Fenêtres

# Console

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

[Recharge de la session précédente restaurée]

> ??"rgl" 3d"
> install.packages("rgl")
[Installation de] install.packages() dans "C:\Users\c.dumont\Documents\R\win-library\2.8"
--- VÉRIFIER LE HASHEM DES FICHES ---
--- SVT PRÉINSTALLER UN MIRROR CRAN POUR CETTE SESSION ---
chargé de l'URL "http://mirror.igmp.fr/pub/CRAN/src/windows/contrib/2.12/rgl_0.104
Content type 'application/zip' length 161794 bytes (154 KB)
URL correcte
downloaded 1.6 Mb

Le package 'rgl' a été décompressé et les sommes MD5 ont été vérifiées avec succès

Les packages téléchargés sont dans
C:\Users\c.dumont\AppData\Local\Temp\RtmpRQnqge\downloaded_packages
>

```

Information

Help files with alias or concept or title matching 'plot 3d' using fuzzy matching:

fields::draps.color	Perspective plot draped with colors in the facets.
rgl::plot.variplot	Plot 3D Variational Variogram
rgl::plot3d	3D Scatterplot
scatterplot3d::scatterplot3d	3D Scatter Plot
stats::plot3d	Plot 3D Scatterplot for an R object

Type "???" to inspect entire "???", or "???" for entire line "???"

Figure: Installation librairie



# Installer et charger une librairie

## Charger une librairie

La commande pour charger une librairie est la suivante:  
*library(NomDeLaLibrairie)*

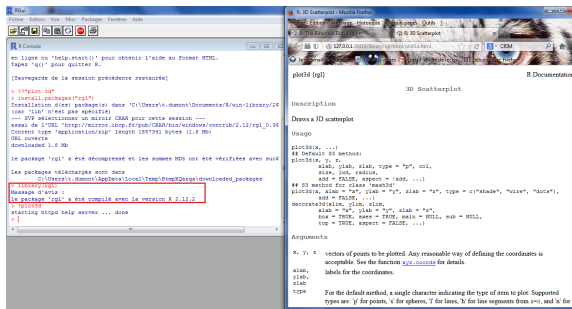


Figure: Charger librairie



# Installer et charger une librairie

## Charger une librairie

La commande pour charger une librairie est la suivante:  
*library(NomDeLaLibrairie)*

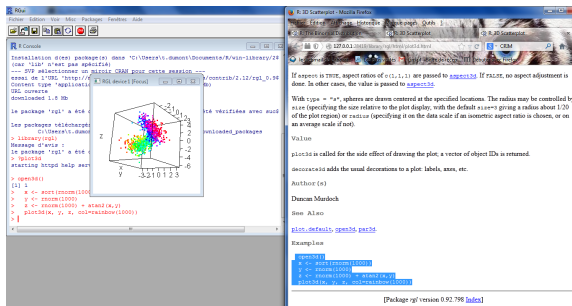


Figure: Charger librairie

ooooooooooooo  
ooo  
ooo

## 1 Introduction

## 2 Formalisme

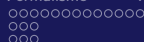
- Les conteneurs
- Opérateurs logiques, boucles et conditions
- Les fonctions

## 3 Probabilités

## 4 L'aide en ligne

## 5 Les librairies

## 6 Enregistrer/charger des données



# Enregistrer des Objets R

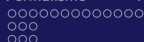
## Sauvegarder Objets R

# Sauvegarder un objet R afin de le reprendre plus tard sous R :

- # Changer le répertoire de travail :  
> *setwd(" C : /Users/.../Donnees")*
- # Sauvegarder l'objet sous l'extension *.Rdata*:  
> *save(ObjetR, file = "nomFichier.RData")*

# Récupérer l'objet :

- # S'assurer que le répertoire de travail est le bon :  
> *getwd()*
- # Changer (si nécessaire) le répertoire courant :  
> *setwd(" C : /Users/.../Donnees")*
- # charger l'objet :  
> *load("nomFichier.RData")*



# Enregistrer des Objets R

## Sauvegarder Objets R

# Pour sauvegarder un objet R dans un autre format : Exemple du fichier .csv (sauvegarde d'une matrice  $M$ )

- # Se placer dans le bon répertoire de travail en utilisant `setwd()`,
- # Sauvegarde de la matrice :  
> `write.csv(M, file = "matrice.csv")`
- # Chargement de la matrice :  
> `M2 = read.csv("Simus.csv", header = TRUE)`

**Attention l'objet  $M_2$  est de type `data.frame`**  
**(`is.data.frame(M2) = TRUE`).**

**Pour le convertir en matrice :**

> `as.matrix(M2)`.



# Quitter R

q()

# Quelques références

## Introduction très complète ...

[http://cran.r-project.org/  
doc/contrib/Goulet\\_introduction\\_programmation\\_R.pdf](http://cran.r-project.org/doc/contrib/Goulet_introduction_programmation_R.pdf)

## Pour démarrer rapidement ...

[http://www.biostat.envt.fr/  
spip/IMG/pdf/Une\\_introduction\\_au\\_langage\\_R.pdf](http://www.biostat.envt.fr/spip/IMG/pdf/Une_introduction_au_langage_R.pdf)

## Pour tout connaitre de R

commandes:

> *help*("nom de la fonction"); ou >?"nom de la fonction";

Si le nom de la fonction n'est pas connue :

>?"terme de Recherche"