INTRODUCTION AU LOGICIEL R

Présentation de R

- R est un système **d'analyse statistique et graphique** créé par Ross Ihaka et Robert Gentleman (1996)
- Développé par des statisticiens « R development Core Team » (21 personnes)
- Version 1.0.0 en 2000, version 2.12.1 en janvier 2011

- R est un **logiciel libre** (License GNU GPL), **http://cran.r-project.org/**
- Gratuit, code source accessible, collaboratif
- Langage de programmation spécifique
- Documentation abondante
- Fonctionne sous Unix, Linux, Windows et Mac

- Possibilités de base extensibles par des "**packages**" (+ de 1800 packages...)
- **Evolue en permanence** car implémentations fréquentes de nouvelles méthodologies statistiques (statistiques spaciales, bayesien,...)
- Utilisé en recherche, en enseignement et en entreprise

Avantages/inconvénients

- **Points forts :**

  - Stockage et manipulation de données
  - Opérateurs pour les tableaux et calcul matriciel, …,
  - Outils intégrés d'analyse de données et d'analyse statistiques
  - Nombreux packages sur les méthodes statistiques récentes,
  - Langage de programmation interprété, simple et efficace,
  - Flexibilité grâce au stockage des résultats sous forme d'objets
  - Possibilité de couplage des fonctions R avec d'autres langages
    (encapsulation de C/Fortran, passerelles Perl/R, Python/R, Matlab/R, …)

- **Points faibles :**

  - Calculs + lents que les langages non interprétés
  - Problème mémoire pour les données trop volumineuses
  - Paramétrisation complexe des options dans l'utilisation des fonctions

Ressources

- **Plus de 80 livres :**

  - Collection spécifique Springer Use R
  - Introductory Statistics with R (Dalgaard)
  - The R book (Crawley)
  - Linear models with R (Faraway)
  - Software for data analysis : programming with R (Chambers)
  - Bayesian Computation With R (Albert)
  - Introducing Monte Carlo Methods with R (Robert & Casella)
  - …

- **Internet:**

  - R home page **http://cran.r-project.org/** (Manuals, Task view, RSiteSearch, Mailing lists, FAQ, ...)
  - R graphics gallery **http://addictedtor.free.fr/graphiques/**
  - **…**

Premiers pas

- **Démarrer et quitter R :**

Windows : double clic sur l'icône Linux : taper R

Quitter : q()

> Apparaît automatiquement au début de chaque ligne

+ apparaît si la ligne précédente est incomplète

- **Aide, démos et exemples :**

```
> help.start()
> help(plot) ou ?plot
> help.search("plot")
   example(fonction)
> apropos("plot")
```

```
> demo()
> demo(library)
>
```

- **Editeur :**

Windows : intégré (tinn-R gratuit) Linux : emacs (ess)

Pour exécuter rapidement un ensemble de lignes avec l'éditeur de texte de R, il suffit de les sélectionner à la souris et de taper Ctr-R ou F5

Les librairies ou packages

- **Installer une librairie :**

  - Windows : Packages -> installer le package -> Choix du site miroir -> choix du package
  - Linux : commande « R CMD install »

- **Charger une librairie :**

  > library() pour connaitre les librairies disponibles dans la session R
  > library(MASS) pour charger le package MASS
  > help(package= “MASS”)

Entrées/sorties

- **Exporter des données :**

```
> write.table(data, file="file.txt", quote=F) #pour un tableau
> write.table(data, file="file.txt") #pour un vecteur
```

- **Exporter des données :**

```
> data = read.table("file.txt",header=T) #pour un tableau
> data = scan("file.txt",header=T) #pour un vecteur
```

- **Exporter des graphiques :**

- Windows : cliquer droit sur le graphe ou "Fichier"- "sauver sous" - "pdf, ps, jpeg" ,...

- Linux : commandes postscript , jpeg, bmp, …

```
> bmp(filename = "graphe1.bmp") # avant l'execution du graphe
> dev.off() # après l'execution du graphe
```

- **Exécuter des commandes contenues dans un fichier :**

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

Opérations élémentaires et objets

- **Opérations élémentaires :**

  - Scalaires : *; -; +; /; ^ ;%% (modulo) Affectation : = ou <-

- **Principaux types :**

  - entier, réel, complexe
  - caractère
  - logique : TRUE, FALSE, NA (not available)

- **Objets de base :**

  - vecteurs, matrices, data.frames, listes, array

  > ls() # retourne la liste des objets de la session. > rm(x) # supprime l'objet x

- **Fonctions is et as :**

  - is.type(x) teste si obj est un objet de type xxx
  - as.type(x) contraint si possible x au type d'objet *type* (complex, real, vector matrix etc...)

```
> x=3                          > is.complex(x)
> is.real(x)                   > as.character(x)
```

Vecteurs et matrices

- **Création de vecteur :**

  - fonction c() : concatène des scalaire ou des vecteurs

```
> x1=c(1,2,3)
> x2 = c(x1,4,5,6)
```

  - seq(from, to, by) ou seq(from, to, length)

```
> x3 = seq(1,6,by=1)
```

  - rep(x,times)

```
> rep(0,10)
> rep(x1,3)
```

- **Création de matrice :**

  - matrix(data,nrow, ncol) : remplit une matrice de taille nrow x ncol colonne par colonne à partir du vecteur data.
  - diag(vect) : crée une matrice diagonale de diagonale le vecteur vect

```
> mat1 = matrix(seq(1,6,by=1),ncol=2)
```

  - cbind(vect1,vect2) : concaténation en colonne

```
> mat2 = cbind(mat1,rep(0,3))
```

  - rbind(vect1,vect2) : concaténation en colonne

Opérations sur les vecteurs et les matrices

- **Extraction d'éléments :**

  - vecteur : vect[i]

```
> x=seq(1,10,by=1)      > x[1]              > x[c(1,3,9)]
> x[-5]                 > x[-c(1,10)]       > x[x>5]
> x[x>9 |x<2]           > x[x<6 & x>3]
> y <- c(NA,x,NA)       > y[ !is.na(y)]
```

  - matrice : mat[i,j]

```
> M=matrix(x,ncol=2)    > M[1,2]          > M[3,]   > M[-1,
```

- **Opérations * + - / :** opérations termes à termes sur des vecteurs ou matrices

```
> y = rep(1,10)         > x+y     > z=c(x,1) > z+y
> u = rbind(x,y)  > u+1
```

- **Calcul matriciel :**

  - produit matriciel : %* %
  - valeurs/vecteurs propres : eigen
  - déterminant : det
  - transposée de la matrice A : t(A)
  - inverse de A : solve(A)
  - solve(A,b) retourne x tel que Ax = b

- **Fonction apply :** apply(X, MARGIN, FUN, ...) applique aux lignes (margin=1) ou colonnes de X (margin=2) de X la fonction FUN

dataframes, listes et array

- **listes :** Objet « fourre-tout » (scalaire, vecteur, matrice, …)

  Accès aux composants d'un objet de type list soit par le nom soit par le numéro entre [[ ]] (très utile pour renvoyer les résultats d'une fonction)

```
> l1=list(a=1,b=seq(1,10,by=1),c=matrix(seq(1,6,by=1),ncol=2))
> l1$a              > l1$b                  > l1$c
```

- **dataframes :** Structure spéciale pour les jeux de données

  - Mêmes opérations que les matrices
  - extraction des colonnes possible comme les listes (avec $)
  - Les colonnes peuvent être de natures différentes (quantitatives et qualitatives)

```
> taille=c(140,160,170)  > yeux =c("bleu","vert","marron")
> H = data.frame(taille,yeux)   > H$taille
```

- **array:** Généralisation du type matrix à plus de 2 dimensions (la fonction apply() reste utilisable)

```
> A = array(1 :12,c(2,3,2))
```

Quelques fonctions utiles

- **Fonctions définie sur des scalaires retournant un scalaire :**

  - sqrt, abs, sin, cos, tan, exp log, log10, …

  - Lorsque le paramètre d'entrée est un vecteur (ou une matrice), la fonction est appliquée sur chacune des composantes.

```
> sqrt(seq(1,6,by=1)    >log(matrix(seq(1,6,by=1),ncol=2))
```

- **Fonctions définie sur des vecteurs ou des matrices :**

  - max, min, sum, prod, length, dim, cumsum, cumprod, sort, diff, unique, …

```
> x=c(seq(1,5,by=1),seq(7,3,by=-1))  > sum(x)  > cumsum(x)
> length(x)                  > sort(x)          > unique(x)
```

- **Fonctions which :**

  - which(vec) retourne les indices des coordonnées du vecteur logique vec qui prennent la valeur TRUE

```
> which(x==5)      > which(x>=5)        > which(x>=5 | x==1)
```

  - cas particulier : which.max et which.min

Ecrire une fonction

- **Structure d'une fonction retournant un objet :**

```
FUN=function(liste_des_paramètres){
commandes
return(objet_retourné) }
> x = FUN(paramètres)
```

- **Structure d'une fonction retournant plusieurs objets :**

```
FUN=function(a,b,c=0){ #affectation de valeurs par défaut
commandes
return(list(noms1=objet1,noms2=objet2) }
> x = FUN(paramètres)
> x1 = x$noms1    > x2 = x$noms2
```

Charger une fonction avec `source("nomfichier.R")`

Structure de contrôle

- **Instruction conditionnelle if :**
  - `if(cond){instr1}`
  - `if (cond){instr1} else{instr2}`

    `> if (x>0) y=log(x) else y=0`
  - `ifelse(cond,instr1,instr2)`

    `> ifelse(x>0,log(x),0)`

- **Itérations :**

  for : `for(variable in sequence) {expr}`

  while : `while(cond) {expr}`

  repeat : `repeat expr break`

```
> for (i in 1:10){
> print(i)}
> i=1
> while(i<11){
> print(i)
> i=i+1}
```

Graphiques

- **Fonction centrale plot :**

```
plot(x, y, ...) # help(plot)
```

Quelques arguments :
type = type="p" (points) ou "l" (ligne) ylim=c(ay,by) et xlim=c(ax,bx) pour fixer la limite des axes
pch : type de points
lty : type de lignes
lwd : épaisseur de la ligne
col : couleur (red, blue, green,…)

Le graphique produit par la fonction plot(x) dépend de la classe de l'objet x.

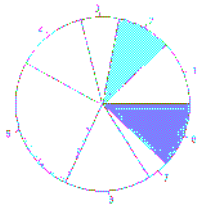
- **Superposition de courbe ou nuages de points avec lines et points :**

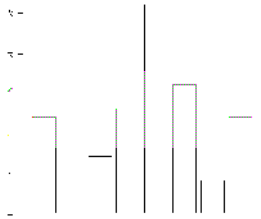
```
lines(x, y)
points(x,y)
abline() # pour tracer des droites
```

Graphiques particuliers

- **Variables qualitatives :**

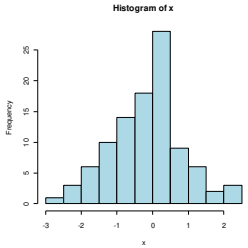
```
pie(x) # diagramme camenbert
barplot(x) # diagramme bâton
```

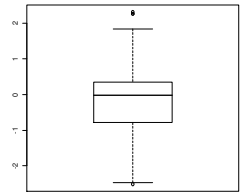


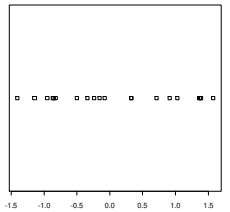


- **Variables quantitatives :**

```
hist(x,nclass) # histogramme de x
boxplot(x) # boite à moustache
stripchart(x)
```



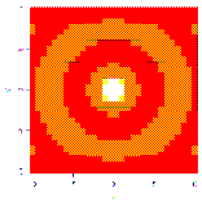


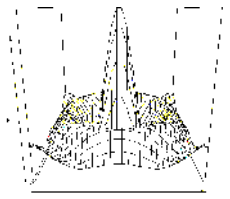


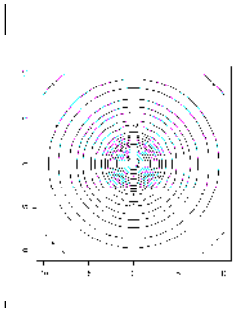
- **Graphiques 3D :**

```
image(x,y,z) # forme d'image
persp(x,y,z) # forme de nappe
contour(x,y,z) # les contours
```

Fonction utile : `z=outer(x,y,fonction)`







- **Tableau et matrice :**

```
pairs(data) # nuage de points colonne par colonne de data
matplot(data) # trace chaque colonne de data
```

Statistiques descriptives

- **Localisation :**

mean(x) Moyenne du vecteur x.
median(x) Médiane du vecteur x.
quantile(x,probs=c(0.1,0.9)) Quantiles à 10% et 90% de x
max(x) Maximum de x
min(x) Maximum de x
which.max(x) Indice du maximum de x
which.min(x) Indice du maximum de x

- **Dispersion :**

sd(x) Ecart type du vecteur x.
var(x) Variance du vecteur x.
IQR(x) Intervalle inter-quartile de x

summary(x) Résumé assez complet de x

- **Forme**

kurtosis(x) Kurtosis de x (library(moments))
skewness(x) Skewness de x (library(moments))

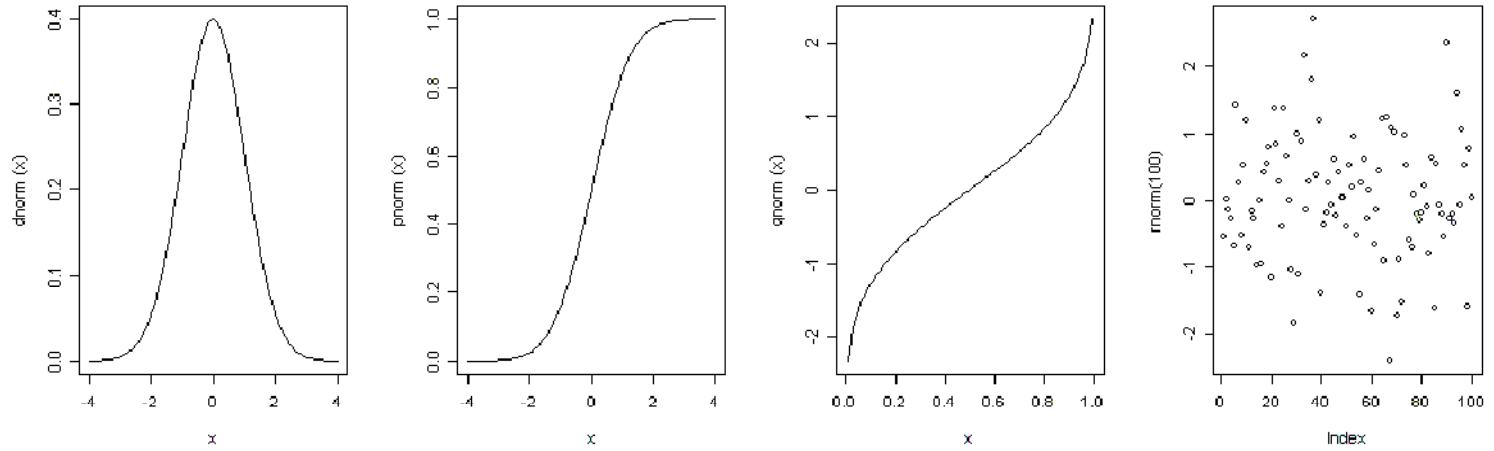
Lois de probabilité

help.search(”Distributions”) Pour voir les distributions existentes

dloi(x,arg1,arg2,...) densité de probabilité de loi(args1,args2,...)
ploi(q,arg1,arg2,...) fonction de répartition de loi(args1,args2,...)
qloi(p,arg1,arg2,...) quantile de loi(args1,args2,...)
rloi(n,arg1,arg2,...) génération aléatoire de n réalisations suivant loi(args1,args2,...)

- **Quelques exemples de loi :**

binom (loi binomiale), norm (loi normale), unif (loi uniforme), lnorm (loi log-normal)
triangle (loi triangulaire library(triangle)) gumbel (loi Gumbel library(evd))
weibull (loi Weibull), …



Ajustement de lois et tests statistiques

- **Estimation des paramètres d'une loi :**

moment(x,order = p, central = FALSE) Moment d'ordre p non centré de x (library(moments))

fitdistr(x,densfun,start) Estimation par maximum de vraisemblance des paramètres de densfun(library(MASS))

density(x,bw) Ajustement non paramétrique de largeur de fenêtre bw

```
> hist(x)
lines(density(x))
```

ecdf(x) fonction de répartition empirique de x

```
> plot(ecdf(x))
```

- **Quelques tests statistiques :**

ks.test(x) test de Kolmogorov-Smirnov
ad.test(x) test de Anderson-Darling (library(ADGofTest))
cramer.test(x,y) test de Cramer Vim Mises pour 2 échantillons (library(cramer))
shapiro.test(x) test de Shapiro-Wilks pour la normalit´e
library(nortest) Divers tests pour la normalit´e

Régression linéaire

- **Définition de la formule :**

```
formule=y~.  modèle linéaire complet.
formule=y~1 # modèle linéaire constant.
formule=y~x1+x2 modèle linéaire avec la constante, x1 et x2.
formule=y~-1+x1+x2 modèle avec x1 et x2 sans la constante.
formule=y~ x1+x2+x1 :x2 # constante, x1, x2 et x1x2.
formule=y~(x1+x2+x3)^2 # constante, x1, x2 et interactions
ordre 2.
formule=y~log(x1) # constante et la variable log(x1).
```

- **Ajustement par moindres carrés :**

```
> fit = lm(formule, data=…)
> fummary(fit)
```

Nombreux résultats attachés à fit et summary(fit) (utiliser names pour les voir)

- **Visualisation**

```
> plot(fit) # divers graphiques sur résidus
> plot(y,predict(fit)) # graphique rée/prédit
> abline(0,1)
```