Introduction langage R
Analyse exploratoire des données

Master M1 IRAD - Université d'Orléans

Christel Vrain
Christel.Vrain@univ-orleans.fr

LIFO (Laboratoire d'Informatique Fondamentale d'Orléans)
Département Informatique - Faculté Sciences
Université d'Orléans

Motivation

- Un monde de données en plein expansion, mais peu d'informations *"data rich but information poor"*
  - Nombreuses bases de données, de plus en plus volumineuses
  - Connexion de bases de données $\Rightarrow$ des vues inattendues
  - L'univers digital contenait approximativement 281 exabytes (281 milliards de gigabytes) en 2007. En 2011, il devrait être 10 fois plus grand.
  - Textes, images, vidéos
- Emergence du "data mining"

Motivation

- Statistique descriptive
- Régression : analyser la relation d'une variable avec d'autres variables
  Exemple : régression linéaire $y = \alpha_0 + \alpha_1 x_1 + \ldots + \alpha_p x_p$
- Fouille de données
  - Classification supervisée
  - Classification non supervisée
  - Prédiction
  - Recherche de motifs fréquents, de règles d'association

Références

- An Introduction to R. W. N. Venables, D. M. Smith and the R Development Core Team
- Enseignements de Statistique en Biologie. A.B. Dufour D. Chessel J.R. Lobry Contributeurs S. Mousset S. Dray. Maintenance système S. Penel
  Site web : http://pbil.univ-lyon1.fr/R/enseignement.html

Langage R

- Langage et environnement pour le calcul statistique et les graphiques
- GNU projet - licence GPL
- Grande variété de techniques statistiques et graphiques : modélisation, classification supervisée et non supervisée, série temporelles, ...

→ gestion des données

→ calcul sur des tableaux

→ outils pour l'analyse de données

→ outils pour la visualisation graphique

→ langage de programmation simple : boucle, récursivité, entrées-sorties

Lancement R

```
R version 2.14.1 (2011-12-22)
Copyright (C) 2011 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
...
R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

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.
...
> 3+5
[1] 8
> log(3)
[1] 1.098612
```


Langage R

- Affectation $<-$ ou $->$ ou *assign*("*x*", *valeur*)

```
> x<-3
> x
[1] 3
> 4->y
```

```
> y
[1] 4
> assign("z",x+y)
> z
[1] 7
```

- Des variables
  - Ne peuvent commencer par un chiffre ou un caractère spécial
  - Sensibles à la casse
  - Des noms courants déjà utilisés

```
> noms<-ls("package:base")
> length(noms)
[1] 1196
```

ls : vecteur de chaînes de caractères donnant les noms des objets dans l'environnement

Structure de données : vecteur numérique

- La structure élémentaire est un vecteur : un nombre est un vecteur

```
> u<-c(1,2,3)
> u
[1] 1 2 3
```

```
> v<-c(u,u)
> v
[1] 1 2 3 1 2 3
```

- $x[i]$ donne le i-ème élément, $x[c(i_1, \ldots, i_j)]$ donne le i_1-ème, ..., i_j-ième élément

```
> v
[1] 1 2 3 1 2 3
> v[4]
[1] 1
```

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

Arithmétique vectorielle

- $+, -, *, \hat{}$, *log*, *exp*, *sin*, *cos*, *sqrt*, ... opérations effectuées point à point
- *min*, *max*, *length*, *range* (vecteur de longueur 2), *sum*, *prod*, *sort*

```
> 2*u
[1] 2 4 6
> u^2+u+1
[1]  3  7 13
> sort(v)
[1] 1 1 2 2 3 3
```

```
> v
[1] 1 2 3 1 2 3
> min(v)
[1] 1
> max(v)
[1] 3
> range(v)
[1] 1 3
```

```
> length(v)
[1] 6
> sum(v)
[1] 12
> prod(v)
[1] 36
> max(u,v, prod(v))
[1] 36
```


Structure de données : vecteur numérique

- D'autres exemples d'arithmétique vectorielle

```
> poids <- c(72, 57, 90, 95)
> taille <- c(1.8, 1.65, 1.9, 1.74)
> imc <- poids/taille^2
> imc
[1] 22.22222 20.93664 24.93075 31.37799
```

- Quand deux vecteurs n'ont pas la même longueur le plus court est répété jusqu'à avoir la longueur de l'autre

```
> mean(imc)
[1] 24.8669
> imc-mean(imc)
[1] -2.64467739 -3.93026049  0.06384832  6.51108956
```

- *NaN* not a number

```
> log(imc-mean(imc))
[1]       NaN       NaN -2.751245  1.873507
Message d'avis :
In log(imc - mean(imc)) : production de NaN
```


Vecteur logique

- Les valeurs sont TRUE, FALSE, NA (Not Available)

```
> v
[1] 1 2 3 1 2 3
> w<-v>2
> w
[1] FALSE FALSE  TRUE FALSE FALSE  TRUE
```

- Les opérateurs logiques sont $<$, $<=$, $>$, $>=$, $==$, $!=$.
- $c1 \& c2$, $c1|c2$, $!c1$
- $is.na(x)$ teste si un objet vaut NA ou NaN
- $is.nan(x)$ teste seulement NaN

```
> x<-log(imc-mean(imc))
Message d'avis :
In log(imc - mean(imc)) : production de NaN
> x
[1]       NaN       NaN -2.751245  1.873507
> x+x
[1]       NaN       NaN -5.502490  3.747014
> is.na(x)
[1]  TRUE  TRUE FALSE FALSE
```


Vecteurs de caractères

- Les éléments d'un vecteur doivent être de même type : logical, numeric, complex, character (ou raw)
- Chaînes de caractères encadrées par des doubles quotes ou des simples quotes (imprimées en double quotes)

```
> x<-0:11
> mode(x)
[1] "numeric"
> x<-as.character(x)
> x
 [1] "0"  "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11"
> mode(x)
[1] "character"
> x<-as.integer(x)
> x
 [1]  0  1  2  3  4  5  6  7  8  9 10 11
```

Index de vecteurs

- un vecteur logique : $a[i]$ où i est un vecteur logique de même longueur que a
  $\rightarrow$ Retourne un vecteur où les valeurs correspondant à l'index FALSE sont omises.

```
> x
[1]       NaN       NaN -2.751245  1.873507
> x[c(FALSE, TRUE, TRUE, TRUE)]
[1]       NaN -2.751245  1.873507
```

```
> x<-x[!is.na(x)]
> x
[1] -2.751245  1.87350
```

- un vecteur de quantités positives
- un vecteur de quantités négatives
- un vecteur de chaînes de caractères : quand un vecteur a un attribut *names* pour identifier ses composantes.

```
> v
[1] 1 2 3 1 2 3
> v[2:5]
[1] 2 3 1 2
> v[-(2:5)]
[1] 1 3
```

```
> fruit<-c(3,2,4)
> names(fruit)<-c("orange","pomme","kiwi"
> fruit[c("orange","kiwi")]
orange   kiwi
     3      4
```


Séquences

Permettent d'engendrer facilement des vecteurs.

- $i : j$ est le vecteur $c(i, i+1, \ldots, j)$ si $i \leq j$, $c(i, i-1, \ldots, j)$ sinon
- La fonction *seq* avec au plus 5 arguments :
  - *from* =
  - *to* =
  - *by* =
  - *length* =
  - *along* = *vecteur* (seul argument si utilisé): crée une séquence de 1 jusqu'à *length*(*vecteur*)

```
> a<-1:5
> a
[1] 1 2 3 4 5
> b<-seq(1,4,0.5)
> b
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0
```

```
> c<-seq(b)
> c
[1] 1 2 3 4 5 6 7
> c<-seq(along=b); c
[1] 1 2 3 4 5 6 7
```


Factor

- Utilisé pour étiqueter des données de vecteurs de même longueur.
  - Quatre individus avec un vecteur donnant leur taille
  - Une étiquette

```
> taille<-c(1.80,1.60,1.65,1.90);taille
[1] 1.80 1.60 1.65 1.90
> et<-c("grand","moyen","moyen","grand")
> etf<-factor(et)
> etf
[1] grand moyen moyen grand
Levels: grand moyen
> moy_taille<-tapply(taille,etf,mean)
> moy_taille
grand moyen
1.850 1.625
```


Fonctions

- Définie par une affectation de la forme
  $$nom \leftarrow function(arg_1, \ldots, arg_n)expression$$
- La valeur retournée par la fonction est la valeur de l'expression
- Conversion de degré Celisus en degré Fahrenheit
  $$degréF = degréC \times \frac{9}{5} + 32$$

```
> x<-c(14,12,13);
> conv<-function(x){x*9/5+32}
> conv
function(x){x*9/5+32}
> conv(x)
[1] 57.2 53.6 55.4
```

Instructions de contrôle

- $expr_1; ...; expr_m$: la valeur du groupe est celle de la dernière expression évaluée
- *if* $(expr_1)$ $expr_2$ *else* $expr_3$:
  - & et | : application sur les vecteurs point à point
  - &&, || : application sur des vecteurs de taille 1, évaluation du second terme seulement si nécessaire
  - *ifelse*(*condition*, *a*, *b*) : retourne un vecteur de la longueur du plus long argument, avec $a[i]$ *condition*$[i]$ est vraie, $b[i]$ sinon
- *for*(*name in* $expr_1$) $expr_2$: *name* est la variable de boucle, $expr_1$ est une expression vecteur (souvent une séquence), $expr_2$ est une expression.

  utilisé bien moins souvent en R
- *repeat expr*
- *while* (*condition*) *expr*
- *break* pour terminer les boucles (repeat)

Tableaux - Matrices

- Un vecteur peut être utilisé comme un tableau à plusieurs dimensions, si on lui associe un vecteur de dimension.

```
> u<-seq(1:30)
> u
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
[23] 23 24 25 26 27 28 29 30
```

```
> dim(u)<-c(5,6);u
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    6   11   16   21   26
[2,]    2    7   12   17   22   27
[3,]    3    8   13   18   23   28
[4,]    4    9   14   19   24   29
[5,]    5   10   15   20   25   30
>
```

```
> dim(u)<-c(6,5);u
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    7   13   19   25
[2,]    2    8   14   20   26
[3,]    3    9   15   21   27
[4,]    4   10   16   22   28
[5,]    5   11   17   23   29
[6,]    6   12   18   24   30
```


Extraction d'éléments ou de sous tableaux

- 2 dimensions $c(6,5)$: ordre $u[1,1]$, $u[2,1]$, ..., $u[6,1]$, $u[1,2]$, ... $u[6,2]$, ...
- 3 dimensions $c(3,5,2)$: ordre $u[1,1,1]$, $u[2,1,1]$, $u[3,1,1]$, $u[1,2,1]$, $u[2,2,1]$, ..., $u[3,5,1]$, $u[1,1,2]$, ..., $u[3,5,2]$

```
> dim(u)<-c(3,5,2);u
, , 1

     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    2    5    8   11   14
[3,]    3    6    9   12   15

, , 2

     [,1] [,2] [,3] [,4] [,5]
[1,]   16   19   22   25   28
[2,]   17   20   23   26   29
[3,]   18   21   24   27   30
```

```
> u[1,1,1]
[1] 1
> u[3,4,2]
[1] 27
> u[3,4,3]
Erreur :
indice
hors limites
> u[c(1,2),1,1]
[1] 1 2
```

```
> v<-u[,,1];v
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    2    5    8   11   14
[3,]    3    6    9   12   15
> w<-u[1,,];w
     [,1] [,2]
[1,]    1   16
[2,]    4   19
[3,]    7   22
[4,]   10   25
[5,]   13   28
```


Tableaux d'indices

- Un tableau peut être construit par la fonction *array*(*vecteur*, *dimension*).
- Une matrice peut servir d'index.
- Arithmétique point à point sur les tableaux

```
> u<-array(1:20, dim=c(4,5));u
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
> i<-array(c(1:3,3:1),c(3,2));i
     [,1] [,2]
[1,]    1    3
[2,]    2    2
[3,]    3    1
> u[i]
[1] 9 6 3
```

```
> u[i]<-0;u
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    0   13   17
[2,]    2    0   10   14   18
[3,]    0    7   11   15   19
[4,]    4    8   12   16   20
> u<-u*u;u
     [,1] [,2] [,3] [,4] [,5]
[1,]    1   25    0  169  289
[2,]    4    0  100  196  324
[3,]    0   49  121  225  361
[4,]   16   64  144  256  400
```

Opération sur les matrices

- Outer product Soient a et b deux matrices et f une fonction à 2 arguments. $outer(a, b, f)$ retourne une matrice de dimension la concaténation des deux dimensions, obtenue en appliquant f sur tous les couples d'éléments x, y avec x dans a et y dans b.

```
> x<-c(3,5)
> y<-c(1,2,3)
> xy <- x %o%y
> xy
     [,1] [,2] [,3]
[1,]    3    6    9
[2,]    5   10   15
```

```
> xy <- outer(x,y,"+")
> xy
     [,1] [,2] [,3]
[1,]    4    5    6
[2,]    6    7    8
```


Listes

- liste : collection ordonnée d'objets, appelés *composants*
- Un composant peut être désigné par
  - son numéro : $List[[i]]$
  - un nom : $List\$name$
  - Attention : $List[i : j]$ retourne une sous liste (avec les noms des composants)

```
> Famille<- list(nom="cv", nbre_enfants=2, age_enfants=c(11,19))
```

```
> Famille
$nom
[1] "cv"
$nbre_enfants
[1] 2
$age_enfants
[1] 11 19

> Famille[[1]]
[1] "cv"‘
```

```
> Famille$nom
[1] "cv"
> Famille$age_enfants[2]
[1] 19
> Famille[[3]][2]
[1] 19
> Famille[["nbre_enfants"]]
[1] 2
> x<-"nom"; Famille[[x]]
[1] "cv"
```

```
> Famille[1:2]
$nom
[1] "cv"

$nbre_enfants
[1] 2

> Famille[1:2]$nom
[1] "cv"
```


Liste suite

- $Lst <- list(name_1 = object_1, ..., name_m = object_m)$
  Les objets de la liste sont copiés (originaux non affectés)
- Extension possible des listes
- Concaténation de listes

```
> Famille$profession <- "ens"
> Famille
$nom
[1] "cv"

$nbre_enfants
[1] 2

$age_enfants
[1] 11 19

$profession
[1] "ens"
```

```
> Identite<-list(nom="cv");
> Enfants<-list(nb_enfants=2)
> Famille<-c(Identite,Enfants)
> Famille
$nom
[1] "cv"

$nb_enfants
[1] 2
```


Data frame

Liste de classe *data_frame* vue comme une matrice avec des colonnes de mode et d'attributs différents.

- Les composants sont des vecteurs, matrices numériques, facteurs, listes, data frames
- Les matrices, listes, data frames apportent leurs composants
- Les vecteurs numériques, logiques, les facteurs sont inclus tels quels, les vecteurs de caractères sont transformés en facteurs

```
> x1<-c(0,1,1,2,3)
> x2<-c(2,1,2.5,0,0.5)
> class<-c(1,1,1,0,0)
> classf<-factor(class)
> data<-data.frame("x1"=x1,"x2"=x2,"classe"=classf)
```

```
> data
  x1  x2 classe
1  0 2.0      1
2  1 1.0      1
3  1 2.5      1
4  2 0.0      0
5  3 0.5      0
```

Lecture de données à partir de fichiers

Les fichiers doivent satisfaire :

- la 1ère ligne doit avoir un *nom* pour chaque variable du data frame.
- chaque ligne doit avoir le numéro du rang suivi des valeurs pour chaque variable
- Par défaut, les valeurs numériques (sauf la ligne) sont lues comme des variables numériques, les autres comme des facteurs. Peut être changé.

```
Fichier famille.txt

Nom Nbre_enfants Age
1 cv 2 c(11,19)
2 lm 2 c(14,16,18)
3 me 2 c(7,11)
```

```
> fa<-read.table("famille.txt",hea
> fa
  Nom Nbre_enfants         Age
1  cv            2    c(11,19)
2  lm            2 c(14,16,18)
3  me            2     c(7,11)
>nfa<-edit(fa)
```