Data Science, Séance 2 : rappels de R

Etienne Côme

21 novembre 2019

R

language de haut niveau

support natif des valeur manquantes

programation objet

éco-système vivant : beaucoup de packages

! plutôt permisif

http://cran.r-project.org/doc/contrib/Lam-IntroductionToR_LHL.pdf

. . .

Les types de base

les vecteurs :

```
# vecteur d'entier
a = c(1,5,10)
class(a)
```

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

```
# de chaine de caractère
b = c("a","g","t","t","c","g","g")
class(b)
```

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

permet de stocker des éléments de même type

opérations de bases c, length, seq, rep, indexation logique

Les types de base

les vecteurs, manipulations de bases :

```
length(a)
```

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

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

```
## [1] 1 5
```

```
i = 1:2;a[i]
```

```
## [1] 1 5
```

```
i = seq(1,length(b),2);b[i]
```

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

```
i = rep(1,5);b[i]
```

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

```
i = rep(c(1,2),5);b[i]
```

Les types de base

les facteurs :

les vecteurs, manipulations de bases :

```r
b = c("a","g","t","t","c","g","g")
c = factor(b,levels=c("a","t","g","c"))
levels(c)
```

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

```r
unclass(c)
```

```
## [1] 1 3 2 2 4 3 3
## attr(,"levels")
## [1] "a" "t" "g" "c"
```

type particulier de vecteurs pour coder des catégories “les niveaux (levels)”

opérations de bases c, length, levels, unclass

! interprétation des chaines de caractères comme des facteurs lors

Les types de base

Les matrices :

```
# matrice d'entier
a = matrix(c(1,5,10,10),2,2)
# de chaine de caractère
b = rbind(c("a","g"),c("t","t"),c("c","g"),c("t","g"))
c = cbind(c("a","g"),c("t","t"),c("c","g"),c("t","g"))
```

permet de stocker des éléments de même type

opérations de bases dim, rbind, cbind, indexation logique

Les types de base

Les matrices :

```r
dim(b)
```

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

```r
t(b)
```

```
##      [,1] [,2] [,3] [,4]
## [1,] "a"  "t"  "c"  "t"
## [2,] "g"  "t"  "g"  "g"
```

```r
dim(t(b))
```

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

```r
a[1,]
```

```
## [1]  1 10
```

```r
b[,2]
```

```
## [1] "g" "t" "g" "g"
```

Les types de base

Les arrays :

```r
# tenseur de dimension 3
a = array(runif(50),dim=c(5,5,2))
a[1,,]
```

```
##             [,1]      [,2]
## [1,] 0.59250264 0.3788759
## [2,] 0.01461666 0.9791246
## [3,] 0.98855426 0.1024477
## [4,] 0.38479225 0.3306272
## [5,] 0.77685344 0.6458938
```

```r
a[,5,]
```

```
##            [,1]      [,2]
## [1,] 0.7768534 0.6458938
## [2,] 0.8098877 0.5233838
## [3,] 0.1544904 0.6300769
## [4,] 0.1188639 0.3751761
```

Les types de base

Les listes :

```
l = list(a,b,c)
length(l)
```

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

```
l[[2]]
```

```
##      [,1] [,2]
## [1,] "a"  "g"
## [2,] "t"  "t"
## [3,] "c"  "g"
## [4,] "t"  "g"
```

```
l = list(a=a,b=b,c=c)
```

permet de stocker des éléments de type différents

opérations de bases length

Les types de base

Les listes :

```
l$c
```

```
##      [,1] [,2] [,3] [,4]
## [1,] "a"  "t"  "c"  "t"
## [2,] "g"  "t"  "g"  "g"
```

```
l[[2]]
```

```
##      [,1] [,2]
## [1,] "a"  "g"
## [2,] "t"  "t"
## [3,] "c"  "g"
## [4,] "t"  "g"
```

permet de stocker des éléments de type différents

opérations de bases length

Les types de base

Les data.frame :

```r
d = data.frame(v1=rep("a",10),v2=1:10,v3=runif(10))
dim(d)
d$v1
d$v4 = factor(rep(c("a","b"),5),levels=c("a","b"))
d[d$v4=="a",]
d[,"v2"]
d[,c(3,1)]
d[,c("v2","v4")]
names(d)
summary(d)
```

permet de stocker des éléments de type différents

= liste de vecteurs només indexable et manipulable comme une matrice

opérations de bases dim, cbind, rbind, names, summary

Les fonctions

```r
f = function(a,b){
  return(a-b)
}
f(5,6)
f(b=5,a=6)
f = function(a=32,b=12){
  a-b
}
f()
f(5,6)
f(b=5,a=6)
```

une variable comme une autre ?

argument nomé et valeur par défaut

pas besoin de return explicite

Les structures de contrôle

```r
for (i in 1:length(a)){}
while(i > 4){i=i-1}
```

! éviter les boucles for, while préférer les opérations vectorielle

```r
a=runif(100000)
t=Sys.time()
for (i in 1:length(a)){a[i]=a[i]+5}
t1=Sys.time()-t
t1
```

```
## Time difference of 0.02120757 secs
```

Version vectorielle

```r
t=Sys.time()
a=a+5
t2=Sys.time()-t
t2
```

```
## Time difference of 0.01061177 secs
```

Quelques fonctions vectorielles

somme (sum), somme cumulée (cumsum), différences finies (diff), max, min . . .

```
a=data.frame(v1=runif(5000),v2=rnorm(5000),v3=rbinom(5000,5
# opération algébrique de base
a$v1+a$v2;a$v1*a$v2;a$v1/a$v2

# produit matriciel
t(a$v1)%*%a$v2

# somme et somme cumulé
sum(a$v2);cumsum(a$v1)

# différence
diff(a$v2)
```

Quelques fonctions vectorielles

somme (sum), somme cumulée (cumsum), différences finies (diff), max, min ...

```
# max, min ...
max(a$v3)
which.max(a$v1)
which(a$v1>0.2)

# concatenation de chaine de caractères
paste(a$v1,a$v2);paste0(a$v1,a$v2)

# sommes sur les matrices
b=matrix(runif(100),10,10)
sum(b);rowSums(b);colSums(b)
```

Apply, lapply, sapply

Appliquer une fonction à chaque élément d’un objet

```r
a=data.frame(v1=runif(5000),v2=rnorm(5000),v3=rbinom(5000,5
# appliquer à chaque lignes
r=apply(a,1,sum)
head(r);class(r);dim(r)

# appliquer à chaque colonnes
r=apply(a,2,function(col){c(max(col),which.max(col))})
r;class(r);dim(r)

# appliquer à tous les éléments d'une liste
b=list(v1=runif(5000),v2=rnorm(5000),v3=rbinom(5000,5,0.2))
r=lapply(b,which.max)
r;class(r)
r=sapply(b,which.max)
r;class(r)
```

à préférer aux boucles. . .

Subset : sample, logical indexing

Sélectionner une partie des données

```r
#logical indexing
a[a$v1>0.98 & a$v3==3,]
```

```
##             v1         v2 v3
## 300  0.9840597  0.1597652  3
## 1811 0.9834785  0.3147051  3
## 2179 0.9881766 -0.4846373  3
## 3595 0.9991476 -1.5718817  3
```

```r
#fonction subset
subset(a,v1>0.98 & v3==3)
```

```
##             v1         v2 v3
## 300  0.9840597  0.1597652  3
## 1811 0.9834785  0.3147051  3
## 2179 0.9881766 -0.4846373  3
## 3595 0.9991476 -1.5718817  3
```

Binning : cut

Prétraiter les variables pour construires des facteurs // intervalles

```
r=cut(a$v2,c(-Inf,-3,-2,2,1,Inf))
class(r);head(r)
```

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

```
## [1] (-2,1] (-2,1] (-2,1] (-2,1] (-2,1] (-2,1]
## Levels: (-Inf,-3] (-3,-2] (-2,1] (1,2] (2, Inf]
```

Jointure : merge, %in%, match

```r
a=data.frame(id=1:500,val1=runif(500))
b=data.frame(id=sample(500,500),val2=runif(500))

# jointure par colonne de même nom
c=merge(a,b)

# recherche des indices de correspondances
match(a$id,b$id)[1:10]
```

```
##  [1] 452  93 475  45 464 144 447 232 168 319
```

```r
# jointure manuelle
d=cbind(a,b$val2[match(a$id,b$id)])
sum(d!=c)
```

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

Jointure : merge, %in%, match

```
# matching multiples
b=data.frame(id=sample(500,1000,replace=T),val2=runif(1000
length(match(a$id,b$id))
```

```
## [1] 500
```

```
length(match(b$id,a$id))
```

```
## [1] 1000
```

```
head(a$id %in% b$id)
```

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

```
length(a$id %in% b$id)
```

```
## [1] 500
```

```
c=merge(a,b)
dim(c)
```

```
## [1] 1000    3
```

Aggrégation : tapply, by, aggregate

```
a=data.frame(id=1:500,val1=runif(500),val2=factor(rbinom(50
aggregate(a$val1,list(a$val2),sum)
```

```
##   Group.1         x
## 1       0 17.631192
## 2       1 62.029426
## 3       2 84.793575
## 4       3 55.891191
## 5       4 29.779820
## 6       5  1.999992
```

Aggrégation : tapply, by, aggregate

```
tapply(a$val1,list(a$val2),summary)
```

```
## $`0`
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.002258 0.193739 0.504010 0.463979 0.729720 0.969246
##
## $`1`
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.003505 0.256001 0.515869 0.492297 0.735279 0.998377
##
## $`2`
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.002896 0.300564 0.508208 0.520206 0.757567 0.995150
##
## $`3`
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.006136 0.210672 0.481605 0.481821 0.742889 0.998449
##
## $`4`
```

Aggrégation : tapply, by, aggregate

```r
by(a$val1,list(a$val2),summary)
```

```
## : 0
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.002258 0.193739 0.504010 0.463979 0.729720 0.969246
## --------------------------------------------------------
## : 1
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.003505 0.256001 0.515869 0.492297 0.735279 0.998377
## --------------------------------------------------------
## : 2
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.002896 0.300564 0.508208 0.520206 0.757567 0.995150
## --------------------------------------------------------
## : 3
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.006136 0.210672 0.481605 0.481821 0.742889 0.998449
## --------------------------------------------------------
## : 4
```

Comptage : table

```r
table(a$val2)
```

```
##
##   0   1   2   3   4   5
##  38 126 163 116  53   4
```

```r
a$val3=rep(c("a","t","g","c"),500/4)
table(a[,c('val2','val3')])
```

```
##     val3
## val2  a  c  g  t
##    0  9  9  7 13
##    1 37 27 27 35
##    2 33 41 51 38
##    3 31 34 28 23
##    4 14 13 11 15
##    5  1  1  1  1
```