

Big Data

TP 1 : Illustration et évaluations

BUT 3

Guillaume Metzler et Antoine Rolland
Institut de Communication (ICOM)
Université de Lyon, Université Lumière Lyon 2
Laboratoire ERIC UR 3083, Lyon, France
guillaume.metzler@univ-lyon2.fr; antoine.rolland@univ-lyon2.fr

Dans le présent TP, on se propose d'illustrer les différents points vus en cours ainsi que certains problèmes réalisés dans les premières fiches de TD.

Dans la suite, nous emploierons le langage  pour la résolution de ces derniers.


Exercice 1 : Retour sur la distance entre individus en grandes dimensions

On se propose d'étudier les distance entre individus en fonction de la dimension du jeu de données.

Une étude a déjà était faite lors du premier TD et on souhaite simplement représenter les résultats graphiquement. Pour cela, on va considérer des échantillons de taille $n = 100$ et des espaces avec les dimensions suivantes :

- $p = 2$
- $p = 10$
- $p = 100$
- $p = 1000$

Les données seront générées selon une loi normale centrée et réduite.

On pourra utiliser les fonctions *dist* pour le calcul des distances entre individus et *hist* ou encore *lines* de  pour représenter les histogrammes des distances.

```
# Génération des données

n = 100
p = 2
M = matrix(runif(n*p), nrow=n, ncol=p)

D = dist(M, method = "euclidean",
```

```

diag = TRUE, upper = TRUE)

p1 <- hist(D,probability = TRUE, col=rgb(0,0,1,1/4), ylim=c(0,2), xlim=c(0,20),
xlab="", main = "Histogrammes")
lines(density(D), col = rgb(0,0,1,1/2), lwd = 3)


n = 100
p = 10
M = matrix(runif(n*p),nrow=n,ncol=p)

D = dist(M, method = "euclidean",
diag = TRUE, upper = TRUE)

p1 <- hist(D,probability = TRUE, col=rgb(1,0,1,1/4), ylim=c(0,2), xlim=c(0,20),
xlab="", main = "Histogrammes", add = TRUE)
lines(density(D), col = rgb(1,0,1,1/2), lwd = 3)


n = 100
p = 100
M = matrix(runif(n*p),nrow=n,ncol=p)

D = dist(M, method = "euclidean",
diag = TRUE, upper = TRUE)

p1 <- hist(D,probability = TRUE, col=rgb(0,0,1,1/4), ylim=c(0,2), xlim=c(0,20),
xlab="", main = "Histogrammes", add = TRUE)
lines(density(D), col = rgb(0,0,1,1/2), lwd = 3)

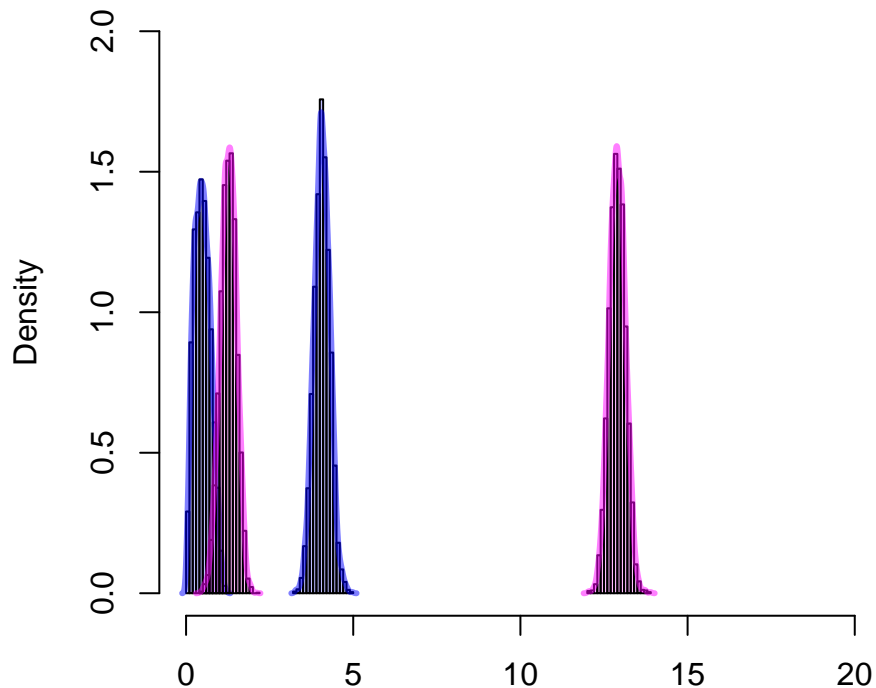

n = 100
p = 1000
M = matrix(runif(n*p),nrow=n,ncol=p)

D = dist(M, method = "euclidean",
diag = TRUE, upper = TRUE)

p1 <- hist(D,probability = TRUE, col=rgb(1,0,1,1/4), ylim=c(0,2), xlim=c(0,20),
xlab="", main = "Histogrammes", add = TRUE)
lines(density(D), col = rgb(1,0,1,1/2), lwd = 3)

```

Histogrammes



Exercice 2 : Retour sur la suite de Fibonacci

On revient sur les approches récursives et itératives permettant de calculer le n -ème terme de la suite de Fibonacci F_n .

1. En reprenant la fiche de TD 4, implémenter la procédure récursive et représenter le temps de calcul en fonction de n (prendre des valeurs de $n \leq 35$).

```
Fibo_recuratif = function(n)
  if (n <= 1){
    return(n)
  } else {
    return(Fibo_recuratif(n-1) + Fibo_recuratif(n-2))
  }

# Evaluation du temps de calcul pour une valeur de n donnée

start.time <- Sys.time()
Fibo_recuratif(30)

## [1] 832040

end.time <- Sys.time()
time_procedure = round(end.time-start.time,2)
```

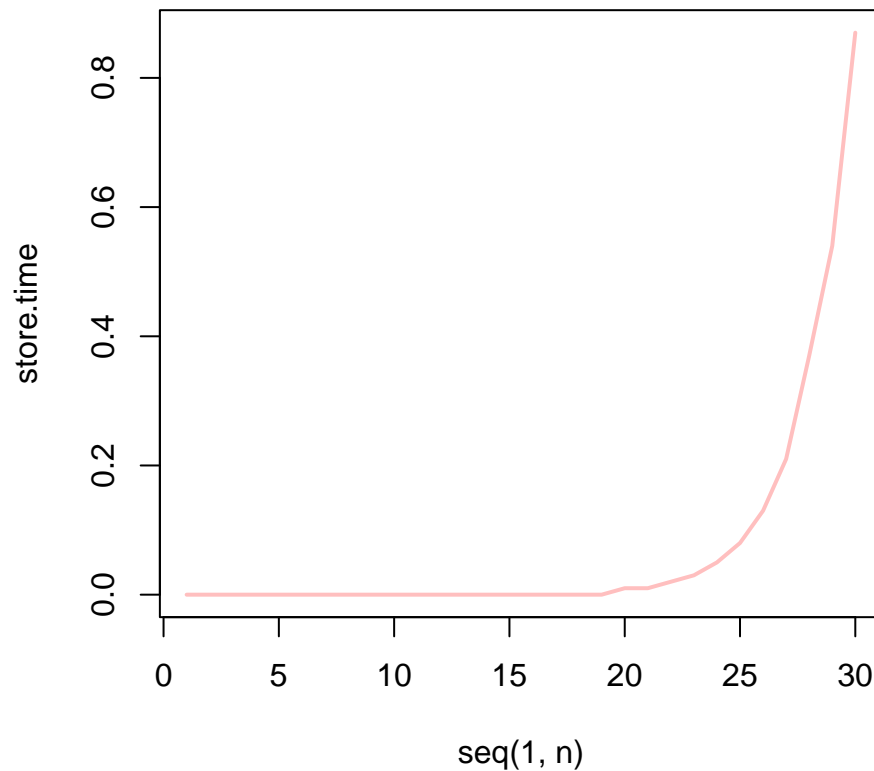
```

# On répète la même chose pour plusieurs valeurs de n
# et on représente la courbe d'évolution du temps de
# calcul en fonction de n

n = 30
store.time = NULL
for (i in 1:n){
  start.time <- Sys.time()
  Fibo_recuratif(i)
  end.time <- Sys.time()
  store.time = append(store.time,as.numeric(round(end.time-start.time,2)))
}

plot(seq(1,n),store.time, type = 'l', col = rgb(1,0,0,1/4), lwd = "2")

```



```

# Pour mettre en évidence le côté exponentielle

# plot(seq(1,n),log(store.time)/log(2), type = 'l', col = rgb(1,0,0,1/4), lwd = "2")

```

2. Calculer le temps nécessaire au calcul de Fibo(30) avec la précédente procédure. Puis, en utilisant le fait (on ne le montrera pas ici) que, pour tout $m > n$, le temps de calcul de Fibo(m) est environ égal au temps de calcul de $Fibo(n) * \left(\frac{1 + \sqrt{5}}{2}\right)^{m-n}$, estimer le temps nécessaire pour calculer :
 - (a) Fibo(50), en heure.

(b) Fibo(500), en années.

```
start.time <- Sys.time()
Fibo_recuratif(30)

## [1] 832040

end.time <- Sys.time()
T_Fibo_30 = as.numeric(round(end.time-start.time,2))

# On pose phi le nombre d'or
phi = (1+sqrt(5))/2

print(paste("Estimation du temps de calcul de Fibo(50) en heures : ",
round(T_Fibo_30 * phi**20 / (60*60) ,2) ))

## [1] "Estimation du temps de calcul de Fibo(50) en heures : 3.7"

print(paste("Estimation du temps de calcul de Fibo(500) en années : ",
round(T_Fibo_30 * phi**470 / (60*60*24*365) ,2) ))

## [1] "Estimation du temps de calcul de Fibo(500) en années : 4.67591903601896e+90"
```

3. Refaire la même chose en implémentant la procédure itérative, de complexité linéaire en temps et constante en mémoire. Mettre en évidence la complexité linéaire en temps.

```
Fibo_iteratif = function(n){
  a = 0
  b = 1
  for (i in 1:n){
    Fib = a+b
    a = b
    b = Fib
  }
  return(Fib)
}

# On répète la même chose pour plusieurs valeurs de n
# et on représente la courbe d'évolution du temps de
# calcul en fonction de n

n = 20000
store.time = NULL
for (i in 1:n){
  start.time <- Sys.time()
  Fibo_iteratif(i)
  end.time <- Sys.time()
  store.time = append(store.time,as.numeric(round(end.time-start.time,6)))
}

plot(seq(1,n),store.time, type = 'l', col = rgb(1,0,0,1/4), lwd = "2", ylim = c(0,0.001
```

