TP bonus : chaines de Markov

Nicolas Jouvin

```r
library(dplyr)
library(ggplot2)
library(knitr)
```

Chaînes de Markov à temps discret : Exercice 50 du polycopié de S

Question 1

Faire l'exercice 50 du polycopié téléchargeable ici

Question 2

Créez la matrice de transition A suivante en R. Vérifiez que c'est bien une matrice stochastique, *i.e* que $\sum_j A_{ij} = 1$.

```
0.1 0.4 0.3 0.2
0.1 0.4 0.3 0.2
0   0.1 0.4 0.5
0   0   0.1 0.9
```

```r
A = matrix(c(0.1, 0.4, 0.3, 0.2,
             0.1, 0.4, 0.3, 0.2,
             0., 0.1, 0.4, 0.5,
             0., 0., 0.1, 0.9) ,ncol = 4, byrow = TRUE )
A
```

```
     [,1] [,2] [,3] [,4]
[1,]  0.1  0.4  0.3  0.2
[2,]  0.1  0.4  0.3  0.2
[3,]  0.0  0.1  0.4  0.5
[4,]  0.0  0.0  0.1  0.9
```

```
rowSums(A) # sanity check
```

```
[1] 1 1 1 1
```

Question 2: Trouver la loi stationaire

On sait que la loi stationaire π vérifie $\pi^\top = \pi^\top A$ et $\sum_i \pi_i = 1$. On va l'estimer par plusieurs méthodes.

2.a) Méthode brute-force

> **Attention**: Ce n'est pas une méthode efficace !

On sait que A^n va converger vers une matrice constante par ligne, avec π en vecteur ligne. Coder une estimation de π en posant $n = 100$.

```
library(expm) # used to compute matrix power
n=100
pi_brute_force = (A %^% n)[1,]
pi_brute_force
```

```
[1] 0.003030303 0.027272727 0.151515152 0.818181818
```

2.b) Via une décomposition en valeur propre et la fonction eigen()

π est le vecteur propre de A^T associé à la v.p. 1, et de norme l_1 unitaire. On va chercher P, D (diagonale), et P^{-1} telles que $A^T = PDP^{-1}$.

A l'aide de la fonction `eigen()` du package **MASS**,

1. retrouver la matrice P (vecteurs propres à droite) et la matrice D (les valeurs propres).
2. vérifier que 1 est bien valeur propre de A^T
3. trouver π

```r
library(MASS)
```

```
Attachement du package : 'MASS'
```

```
L'objet suivant est masqué depuis 'package:dplyr':

    select
```

```r
# Get the eigenvectors of A, note: R returns right eigenvectors
r=eigen(t(A))
# The right eigenvectors
P = r$vectors
# The eigenvalues
vp <-r$values
# Sanity check on the spectral decomposition: we should get t(A) back
t(P%*%diag(vp)%*%ginv(P))
```

```
              [,1]          [,2] [,3] [,4]
[1,]  1.000000e-01  4.000000e-01  0.3  0.2
[2,]  1.000000e-01  4.000000e-01  0.3  0.2
[3,] -1.580376e-16  1.000000e-01  0.4  0.5
[4,] -1.827045e-16 -2.874056e-16  0.1  0.9
```

```r
vp # 1 is an eigenvalue of A^T
```

```
[1] 1.000000e+00 5.732051e-01 2.267949e-01 7.260434e-17
```

```r
pi_eigen = P[,1] # On extrait le vecteur propre de A^T (à droite) correspondant:
pi_eigen # le vp normalisé en norme l2
```

```
[1] -0.003639807 -0.032758259 -0.181990327 -0.982747765
```

```r
# Se serait-on trompé ? **Non**, c'est simplement que les vecteurs propres sont défini à u
pi_eigen = pi_eigen / sum(pi_eigen)
pi_eigen
```

```
[1] 0.003030303 0.027272727 0.151515152 0.818181818
```

Comparer le π trouvé à celui de la méthode "brute-force" en norme l_1.

```r
sum(abs(pi_eigen - pi_brute_force)) # Les 2 méthodes trouvent le même résultat !
```

```
[1] 5.132613e-15
```

2.c) efficacité des 2 méthodes

La seconde méthode est plus général que la première car, avec une décomposition en valeur propre on peut calculer A^n pour tout n

$$(A^T)^n = PD^nP^{-1} \iff A^n = (PD^nP^{-1})^T$$

Coder A^n pour $n = 10^5$ en **R** de manière efficace.

```r
n = 1e5
statio_mat = t(P %*% diag(vp^n) %*% ginv(P)) # n'oubliez pas de transposer
statio_mat
```

```
            [,1]       [,2]      [,3]      [,4]
[1,] 0.003030303 0.02727273 0.1515152 0.8181818
[2,] 0.003030303 0.02727273 0.1515152 0.8181818
[3,] 0.003030303 0.02727273 0.1515152 0.8181818
[4,] 0.003030303 0.02727273 0.1515152 0.8181818
```

Question 3: Système linéaire

Retrouver π en resolvant le système linéaire sur-contraint comme vu dans les slides.

Question 4: simulation de la chaîne

On se donne un $X_0 \in \{0, \dots, 3\}$ et un temps d'arrêt n. On souhaite simuler une chaine $(X_0, \dots, X_n)$

Coder la fonction `sample_chain(n, x_0)` en **R**.

```r
sample_chain = function(n, x_0) {
  X = rep(0, n+1) # allocate a vector of size n+1
  X[1] = x_0
  for (i in 2:(n+1)) {
    idx = X[i-1] + 1 # indexation of states begin at 0, but R begins at 1 !
    X[i] = sample(0:3, size = 1, prob = A[idx,])
  }

  return(X)
}

a_run = sample_chain(n=10, x_0=0)
a_run
```

```
[1] 0 1 3 3 3 2 2 3 3 3 3
```

Question 5: Ergodicité

Avec n assez grand on s'attend à ce que la chaîne "oublie son passé" et visite les états conformément à la distribution stationnaire.

Proposez une petite simulation pour vérifier cela. Essayer différentes valeur de x_0 en point de départ.

```r
x_0 = 3
n = 2e4
big_run = sample_chain(n, x_0)
knitr::kable(table(big_run) / n)
```

big_run	Freq
0	0.00290
1	0.02480
2	0.14495
3	0.82740

```r
#Recall the stationary distribution
knitr::kable(pi_eigen)
```

x
0.0030303
0.0272727
0.1515152
0.8181818

```r
# One can also plot the histogram of the frequency
hist(big_run, probability = TRUE)
```