TD estimateur de Bayes & loi stationaire

Nicolas Jouvin

2022-09-13

Estimateur de Bayes pour le modèle multinomial avec a priori de Dirichlet

On rappelle le modèle :

1. $\pi \sim \mathcal{D}_K(\alpha_1, \dots, \alpha_K)$
2. $x \mid \pi \sim \mathcal{M}_K(N, \pi)$

On souhaite comparer empiriquement les estimateurs du maximum de vraisemblance $\hat{\pi}_{ML,l} = x_k/N$ et de Bayes $\hat{\pi}_{Bayes,k} = \alpha_k + x_k/(\sum_l \alpha_l + x_l)$ pour la perte l_2.

On fixe $K = 4$ et les valeurs de paramètres suivantes, mais votre code doit pouvoir se généraliser facilement.

```
K = 4
alpha = c(1,1,1,1)
PI = c(0.1, 0.2, 0.65, 0.05)
stopifnot(sum(PI) == 1) # sanity check
```

Question 1

Coder une fonction `simu_multi(N, PI)` qui simule un modèle multinomial avec paramètre N et π. Indice : voir `?rmultinom`

```
library(stats)


simu_multi = function(N, PI) {
  x = rmultinom(n=1, size = N, prob = PI)
  return(x)
}
```

Question 2

Soit la simulation suivante:

```
a_simu = simu_multi(N=100, PI=PI)
a_simu
```

```
##      [,1]
## [1,]   11
## [2,]   12
## [3,]   73
## [4,]    4
```

Coder les 2 estimateurs $\hat{\pi}_{ML}$ et $\hat{\pi}_{Bayes}$ en **R** puis calculer leur différence absolue (norme L1 de la différence) $\|\hat{\pi}_{ML} - \hat{\pi}_{Bayes}\|_1$

```r
PI_ML = a_simu / sum(a_simu)
PI_Bayes = (a_simu + alpha) / sum(a_simu + alpha)
sum(abs(PI_ML - PI_Bayes))
```

```
## [1] 0.03692308
```

Question 3: comparaison des performances par simulation

On souhaite faire un nombre grand d'expériences (disons 100) pour des valeurs de N qui varient, afin de mieux comprendre la différence et la similarité entre les 2 estimateurs au fur et à mesure que le nombre d'observations augmente.

```r
n_exp = 100
Ns = c(seq(10,1000, 10), 1e4)
```

En **R**, pour chaque valeurs de N dans le vecteur `Ns`, répéter `n_exp` expériences de simulation selon le modèle multinomiale avec N et $\pi^\star$. Calculer la valeur moyenne des trois erreurs suivantes

1. Erreur relative: $\|\hat{\pi}_{ML} - \hat{\pi}_{Bayes}\|_1$
2. Erreur de l'estimateur frequentiste : $\|\hat{\pi}_{ML} - \pi^\star\|_1$
3. Erreur de l'estimateur Bayésien : $\|\hat{\pi}_{Bayes} - \pi^\star\|_1$

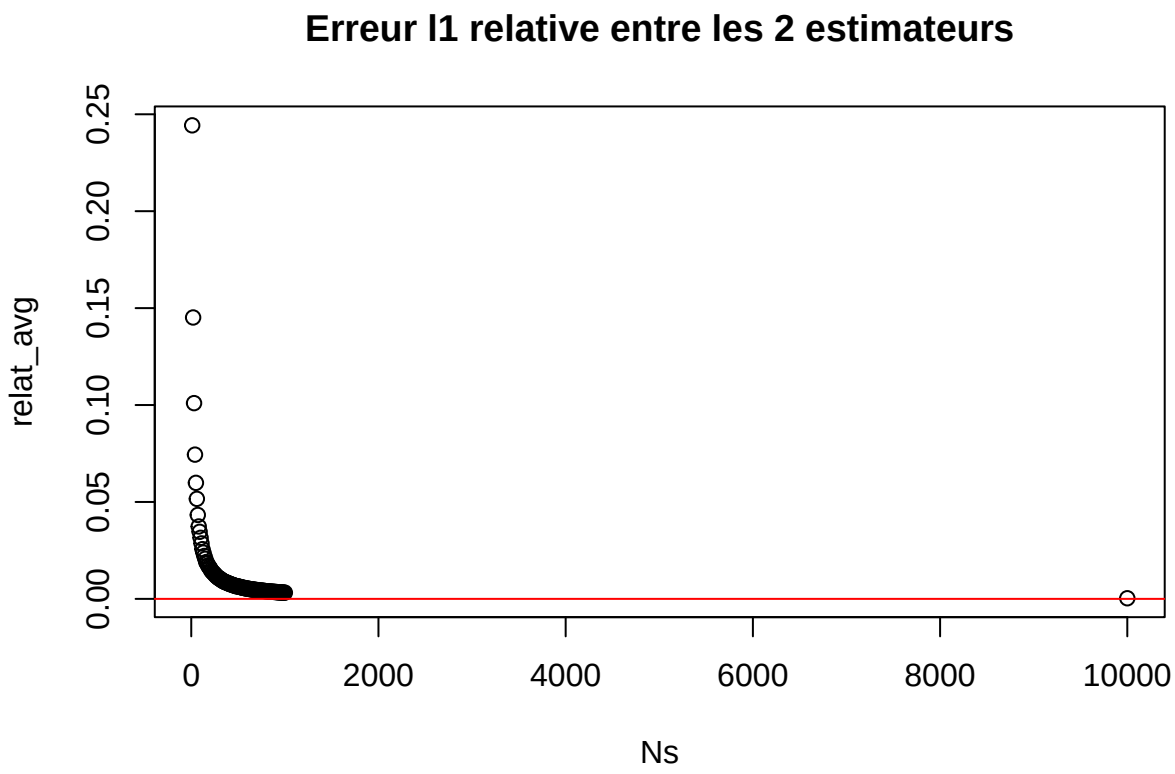
```r
n_exp = 100
Ns = c(seq(10,1000, 10), 1e4)

l1_relat = matrix(0, n_exp, length(Ns))
l1_ML = matrix(0, n_exp, length(Ns))
l1_Bayes = matrix(0, n_exp, length(Ns))

for (j in 1:length(Ns)) {
  N = Ns[j]
  for (i in 1:n_exp) {
      a_simu = simu_multi(N=N, PI=PI)
      PI_ML = a_simu / sum(a_simu)
      PI_Bayes = (a_simu + alpha) / sum(a_simu + alpha)
      l1_relat[i, j] = sum(abs(PI_ML - PI_Bayes))
      l1_ML[i, j] = sum(abs(PI_ML - PI))
      l1_Bayes[i, j] = sum(abs(PI_Bayes - PI))

  }
}

relat_avg = colMeans(l1_relat) # average each column to get average for the 100 expes
ML_avg = colMeans(l1_ML)
Bayes_avg = colMeans(l1_Bayes)
```

Question 4 : visualisation

Tracer 2 graphiques en **R** :

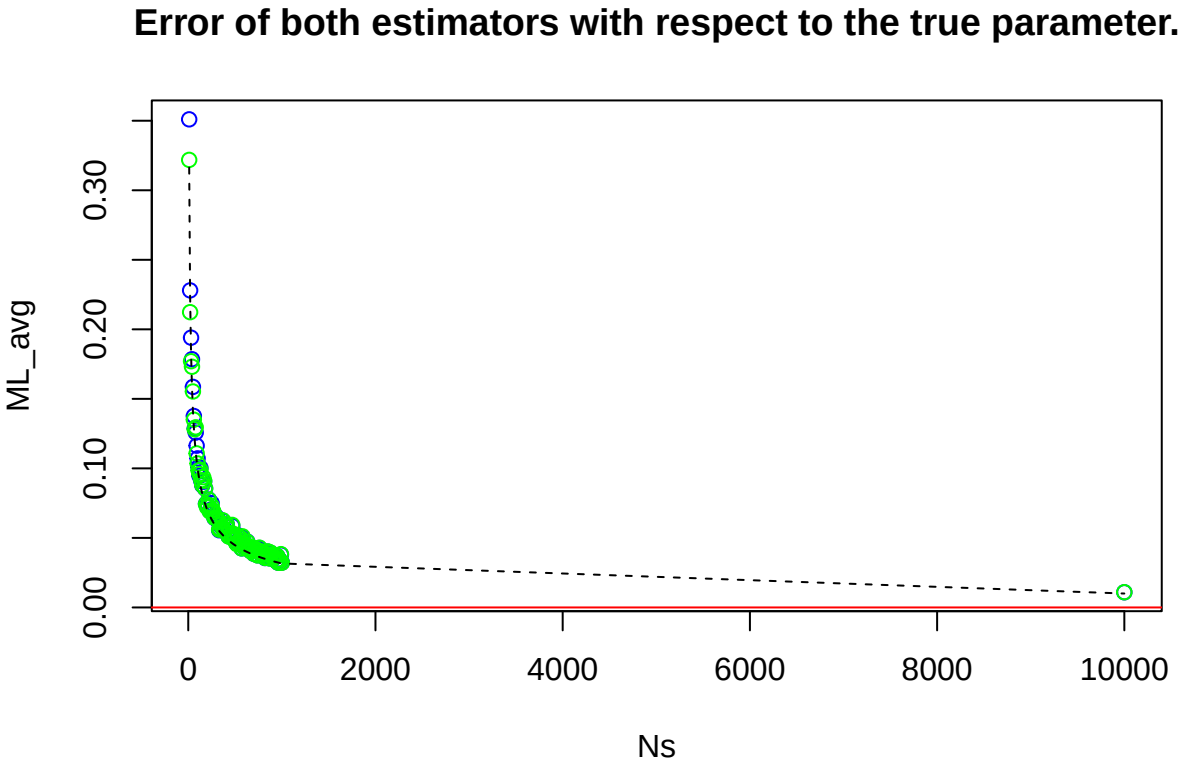
1. l'évolution de l'erreur relative entre les 2 estimateurs avec N

```r
plot(Ns, relat_avg, main = "Erreur l1 relative entre les 2 estimateurs")
abline(h=0, col='red')
```



2. l'évolution avec N de l'erreur d'estimation par rapport à $\pi^\star$ pour les deux estimateurs sur le même graphique.

```
plot(Ns, ML_avg, col='blue', main = "Error of both estimators with respect to the true parameter.")
points(Ns, Bayes_avg, col='green')
lines(Ns, 1/ sqrt(Ns), lty=2) # expected speed of convergence with N
abline(h=0, col = "red")
```



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
```

```
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

```
library(MASS)
# 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
```

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

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

```
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
```

```
# Se serait-on trompé ? **Non**, c'est simplement que les vecteurs propres sont défini à une constante
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: 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 2 1 2 2 2 3 3 3 3 3
```

Question 4: 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	0.00270
1	0.02715
2	0.15000
3	0.82020

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

x
0.0030303
0.0272727
0.1515152
0.8181818

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