TP3 Statistiques : Estimation du maximum de vraisemblance et l'intervalle de confiance

21 mars 2025

Le modèle d'un paramètre: La loi Bernoulli

Soit $X_1, \ldots, X_n$ un échantillon de Bernoulli $\mathcal{B}(p)$ avec $p = 0.6$. Nous voulons estimer p à partir d'une réalisation de cet échantillon $x = (x_1, \ldots, x_n)$. de X

Ex1. Simuler un échantillon i.i.d de taille $n = 10$ en utilisant `rbinom`.

```
n <- 10
p <- 0.6
x <- rbinom(n = n, size = 1, prob = p)
```

Quelle est une estimation simple de p ?

$\hat{p} = \frac{1}{n} \sum_{i=1}^{n} x_i$

Ex2. Créer une fonction de vraisemblance, nommée `L_bern`, en fonction de (p, x), qui donne la vraisemblance d'un échantillon $x = (x_1, \ldots, x_n)$ pour une valeur donnée de p.

```
L_bern <- function(p, x){## vraisemblance
  return(prod(p**x * (1 - p)**(1 - x)))
}
```

Ex3.. Pour l'échantillon généré dans l'**Ex1**, calculer la vraisemblance de cet échantillon des lois Bernoulli de paramètres p allant de 0 à 1. Tracer la courbe des valeurs calculées. Que remarquez-vous?

```
pvec = seq(0, 1, by = 0.01)
Lp = sapply(pvec, L_bern, x = x)
Lp
plot(pvec, Lp)
```

La vraisemblance est maximale pour p = 0,6.

Ex4.. En utilisant la fonction `optim` de R, trouvez la valeur de p la plus probable d'avoir généré cet échantillon.

```
?optim  ### la description de l'optim
```

```
mL_bern <- function(p,x){ -L_bern(p,x) }
## optimization standard (minimization)
p0 = 0.5 # valueur initiale pour l'algorithme
res = optim(p0, mL_bern, x=x, method = "BFGS")
res
```

Attention : `optim` est par défaut une routine de **minimization**. Avec la méthode de *L-BFGS-B* dans la fonction `optim`, vous pouvez traiter des contraintes sur le(s) paramètre(s), lorsque qu'il est nécessaire.

```
## optimization avec contraintes
pmin = 0; pmax = 1
```

```
res1 = optim(p0, mL_bern, x=x, method="L-BFGS-B", lower=pmin, upper=pmax)
res1
```

Ex5. Tester avec des échantillons de taille n allant de $n = 10$ à $n = 2000$ et comparer l'écart entre la valeur théorique attendue et la valeur obtenue. Que remarquez-vous? Comment combattre l'instabilité numérique due aux multiplications de probabilités?

```
size <- seq(10,2000, by=10)
para <- rep(0, length(size))

for(k in seq_along(size))
{
  x<-rbinom(size[k],1,p)
  pmin<-0
  pmax<-1
  res2 <- optim(p0, mL_bern, x=x, method="L-BFGS-B", lower=pmin, upper=pmax)
  para[k]<-res2$par
}
plot(size, para)
```

Utiliser la log-vraisemblance au lieu de la vraisemblance:

```
log_L <- function(p,x){
  return(-log(p) * sum(x) - log(1 - p) * sum(1 - x))
} #log vraisemblance
size <- seq(10,2000, by=10)
para <- rep(0, length(size))
for(k in seq_along(size))
{
  x<-rbinom(size[k],1,p)
  pmin<-0.01
  pmax<-0.99
  res2 <- optim(p0, log_L, x=x, method="L-BFGS-B", lower=pmin, upper=pmax)
  para[k]<-res2$par
}
plot(size, para)
```

Ex6. Donner les intervalles de confiance de niveau 80% pour le paramètre p, à partir de la loi théorique asympotique.

En utilisant que $p(1-p) \leq 1/4$, on obtient que l'intervalle de confiance de niveau 80% pour le paramètre p est $\left[\bar{X} - \frac{q_{1-\alpha/2}}{2\sqrt{n}}, \bar{X} + \frac{q_{1-\alpha/2}}{2\sqrt{n}}\right]$ avec $\alpha = 0.2$ et $q_{1-\alpha/2}$ le quantile d'ordre $1-\alpha/2$ de la loi $N(0,1)$.

```
alpha = 0.2
q = qnorm(1 - alpha / 2, mean = 0, sd = 1)
```

On peut aussi utiliser la loi de Student.

Alternativement, donner les intervalles de confiance d'après l'inégalité de Bienaymé-Tchebychev et l'inégalité de Hoeffding.

L'intervalle de confiance de niveau 80% pour le paramètre p avec Bienaymé-Tchebychev est $\left[\bar{X} - \frac{1}{2\sqrt{n\alpha}}, \bar{X} + \frac{1}{2\sqrt{n\alpha}}\right]$ avec $\alpha = 0.2$.

L'intervalle de confiance de niveau 80% pour le paramètre p avec Hoeffding est $\left[\bar{X} - \sqrt{-\frac{\log(\alpha/2)}{2n}}, \bar{X} + \sqrt{-\frac{\log(\alpha/2)}{2n}}\right]$ avec $\alpha = 0.2$.

Ex7. Pour n fixé, simuler 500 échantillons et donner des intervalles de confiance correspondants. Vous pouvez faire varier la taille de l'échantillon et le paramètre réel. Comptez le nombre de fois où l'intervalle de confiance contient le vrai paramètre. Quelle est votre conclusion?

```
p <- 0.6
n <- 100
m <- 500
############################################################
#les intervalles de confiance à partir de la loi théorique asympotique
low_int_asym <- rep(0, m)
upp_int_asym <- rep(0, m)
sum_asym <- 0

for (i in 1:m){
  x <- rbinom(n, 1, p)
  x_bar <- sum(x) / n
  low_int_asym[i] <- x_bar - q / (2 * sqrt(n))
  upp_int_asym[i] <- x_bar + q / (2 * sqrt(n))
  sum_asym <- sum_asym + (low_int_asym[i] <= p) * (upp_int_asym[i] >= p)

}
## 500 intervalles de confiance correspondants aux 500 échantillons simulés
low_int_asym
upp_int_asym

##le nombre des intervalles de confiance contient le vrai paramètre pour chaque méthode
sum_asym

#########################################################
#les intervalles de confiance d'après l'inégalité de Bienaymé-Tchebychev
low_int_BT <- rep(0, m)
upp_int_BT <- rep(0, m)
sum_BT <- 0

for (i in 1:m){
  x <- rbinom(n, 1, p)
  x_bar <- sum(x) / n
  low_int_BT[i] <- x_bar - 1 / (2 * sqrt(0.2 * n))
  upp_int_BT[i] <- x_bar + 1 / (2 * sqrt(0.2 * n))
  sum_BT <- sum_BT + (low_int_BT[i] <= p) * (upp_int_BT[i] >= p)
}
## 500 intervalles de confiance correspondants aux 500 échantillons simulés
low_int_BT
upp_int_BT

##le nombre des intervalles de confiance contient le vrai paramètre pour chaque méthode
sum_BT

#############################################################
#les intervalles de confiance d'après l'inégalité de Hoeffding
low_int_H <- rep(0, m)
upp_int_H <- rep(0, m)
sum_H <- 0
for (i in 1:m){
```

```
  x <- rbinom(n, 1, p)
  x_bar <- sum(x) / n
  low_int_H[i] <- x_bar - sqrt(-log(0.2 / 2) / (2 * n))
  upp_int_H[i] <- x_bar + sqrt(-log(0.2 / 2) / (2 * n))
  sum_H <- sum_H + (low_int_H[i] <= p) * (upp_int_H[i] >= p)

}
## 500 intervalles de confiance correspondants aux 500 échantillons simulés
low_int_H
upp_int_H

#le nombre des intervalles de confiance contient le vrai paramètre
sum_H
```

L'intervalle de confiance asymptotique semble être plus précis que les autres.

Un modèle à deux paramètres: La loi Beta

Soit $X_1, \ldots, X_n$ un échantillon de n variables indépendantes de loi de Beta ($\theta = (\alpha, \beta)$). Simuler un échantillon i.i.d de taille $n = 50$ avec $\theta_0 = c(2, 3)$.

```
n<-500
theta<-c(2, 3)
X <- rbeta(n, theta[1], theta[2])
head(X)
```

Ex8. Présenter l'histogramme des données simulées. Choisir trois paramètres candidats, disons, θ_0 (vrai) θ_1, θ_2. Comparer l'histogramme avec les densités candidates. Que remarquez-vous?

```
theta_0 <- theta
X<- rbeta(n=n, theta_0[1], theta_0[2])
theta_1 <- c(5, 2)
theta_2 <- c(1, 4)
xx<- seq(from = 0, to = 1, by=0.01)
hist(X,, probability = TRUE, main ="Histogramme de X")
lines(xx, dbeta(xx, theta_0[1], theta_0[2]),col="green")
lines(xx, dbeta(xx, theta_1[1], theta_2[2]),col="red")
lines(xx, dbeta(xx, theta_2[1], theta_2[2]),col="blue")
legend("topright", legend=c("Beta(2, 3)","Beta(5, 2)","Beta(1, 4)"),col = c("green","red","blue"),lty =
```

L'histogramme semble proche de la loi avec le vrai paramètre.

Ex9. Générer une fonction de la log-vraisemblance avec les arguments (θ, x), qui donne la log-vraisemblance d'un échantillon $x = (x_1, \ldots, x_n)$ pour une valeur donnée de $\theta = (\alpha, \beta)$. Pour votre échantillon, évaluer la log-vraisemblance de paramètre $\theta = (\alpha, \beta)$, en faissant varier un paramètre à la fois, et visualizer la fonction sur $a_0 \leq \alpha \leq a_1, b_0 \leq \beta \leq b_1$ pour votre choix appropriés de limites. Que remarquez-vous?

Indication : Pour plus de simplicité, il suffit de tracer la tranche des courbes unidimensionnelle en supposant que l'autre est fixe: $\ell(\alpha|\beta = \beta_0)$ pour plusierus valeurs de β_0 et $\ell(\beta|\alpha = \alpha_0)$ pour plusieurs valeurs de α_0.

```
logL_beta <- function(theta,x){
  return(length(x) * log(gamma(sum(theta)) / (gamma(theta[1]) * gamma(theta[2])))
   +(theta[1] - 1) * sum(log(x)) +  (theta[2] - 1) * sum(log(1 - x)))
} #log vraisemblance
aa <- seq(from = 0.1, to = 3, by=0.01)
bb <- rep(3, length(aa))
vtheta =  matrix(c(aa, bb), ncol = 2, byrow = FALSE)
```

```
Lp = apply(vtheta, 1, logL_beta, x=X)
plot(aa, Lp)
```

```
bb <- seq(from = 0.1, to = 5, by=0.01)
aa <- rep(2, length(bb))
vtheta =  matrix(c(aa, bb), ncol = 2, byrow = FALSE)
Lp = apply(vtheta, 1, logL_beta, x=X)
plot(bb, Lp)
```

Nous pouvons également utiliser des visualisations 2-dim ou 3-dim avec `contour` ou `image`. Une visualisation plus avancée peut être réalisée à l'aide du mudule (`plot3D`). Voici un exemple de code.

```
aa = seq(0.5, 5, by=0.2)
bb = seq(0.5, 6, by=0.2)
vtheta = expand.grid(a=aa, b=bb); vtheta

# evaluer la (negative) vraisemblance
mlogL_beta = function(theta, x){
  return(-logL_beta(theta, x))
}
log_lik = apply(vtheta, 1, mlogL_beta, x=X)
cbind(vtheta, log_lik)
# chercher le minimum
iopt = which.min(log_lik)
a1 = vtheta[iopt,1]
b1 = vtheta[iopt,2]
## transformer en matrice
zmat = matrix(log_lik, nrow=length(aa), ncol=length(bb))
rownames(zmat) = aa
colnames(zmat) = bb
## contour plots
?contour
contour(aa, bb, zmat)
points(a1, b1, pch=8, col=2) ## ajouter les points minimum
```

Ex10. Donner l'expression mathématique du vecteur Score (les dérivées premières) à laquelle l'EMV répond et trouver l'information de Fisher. En utilisant la fonction `optim`, trouver l'estimateur du maximum de vraisemblance. Quelle est la loi asymptotique d'estimateur?

```
library(rootSolve)
#vecteur Score (les dérivées premières du log-vraisemblance)
Score = function(theta,x){#On rappelle que theta=(alpha, beta)
  z=numeric()
  #dérivées premières en theta[1](ou alpha)
  z[1]= log(prod(x))-length(x)*(-digamma(theta[1]+theta[2])+digamma(theta[1]))
  #dérivées premières en theta[2](ou beta)
  z[2]= log(prod(1-x))-length(x)*(-digamma(theta[1]+theta[2])+digamma(theta[2]))
  return(z)
}
#l'information de Fisher
Fisher = function(theta){
  x = X
  return(mean((Score(theta,x))**2))
}
```

```r
#EMV
##par chercher les zéros du Score, en utilisant "library(rootSolve)" avec la command "multiroot"
theta_hat = multiroot(f=Score,c(1.8,2.5), x=X)$root
theta_hat
##par trouver le minimum du (negative) log-vraisemblance, en utilisant la command "optim"
theta_tilda = optim(c(1,5),mlogL_beta,x=X)$par
theta_tilda
```

Ex11. Construire des intervalles de confiance asymptotique de niveau 0.90. Incluent-ils les vrais paramètres ? Tester avec des échantillons de taille $n = 20$ et $n = 100$. Quel est l'effet de la taille de l'échantillon ?

```r
#l'intervalle de confiance de niveau 90% de la moyenne

##input taille de l'échantillon, sa moyenne et son écart-type
n <- c(20,100)
x <-list(rbeta(n[1],2,3),rbeta(n[2],2,3))
xbar <- c(mean(x[[1]]),mean(x[[2]]))
s <- c(sd(x[[1]]),sd(x[[2]]))
##Vrai valeur de la moyenne
m <- 2/5       # = alpha/(alpha+beta)
##calculer la marge d'erreur
margin <- qt(0.95,df=n-1)*s/sqrt(n) # en utilisant la loi de Student

##calculer les bornes inférieures et supérieures de l'intervalle de confiance de la moyenne
low <- xbar - margin
low

high <- xbar + margin
high

#La longueur de l'intervalle diminue quand n augmente.
```

La longueur de l'intervalle diminue quand n augmente.

Ex12. Répéter l'estimation 100 fois avec de nouveaux ensembles de données et tracer les valeurs estimées $\hat{\alpha}$ vs $\hat{\beta}$. Sont-elles indépendantes ? Visualisez vos résultats à l'aide d'un histogramme. Ajoutez la densité de la loi asymptotique. Que remarquez-vous?

```r
n=100
theta_loop = matrix(0,n,2)
for (i in 1:100){
  x = rbeta(n,2,3)
  theta_loop[i,] = optim(c(1,5),mlogL_beta,x=x)$par
}

hist(theta_loop[,1],freq=FALSE,xlab='alpha',main = "Histogramme de l'estmation de alpha")

hist(theta_loop[,2],freq=FALSE,xlab='beta',main = "Histogramme de l'estmation de beta")
```

Ex13. Réalisez une étude de simulation pour évaluer la performance des intervalles de confiance et résumez vos conclusions.