



# TP 3 – SY02

## Estimation et théorème central limite



Les questions/sections marquées par un  sont des questions plus ouvertes qui peuvent être abordées dans un deuxième temps.



Les questions/sections marquées par un  sont des questions qui sont prévues pour être traitées en autonomie en dehors de la séance de TP.

### 1 Estimation par la méthode des moments

On modélise un phénomène par une variable aléatoire  $X$  suivant une loi uniforme sur l'intervalle  $[0, a]$  avec  $a$  entier. Afin d'estimer le paramètre  $a$ , on cherche un estimateur avec la méthode des moments. On utilisera la fonction suivante pour générer un échantillon de taille  $n$  :

```
runifa <- function(n) {  
  if(!exists("param")) param <- sample(10:20, 1)  
  runif(n, min = 0, max = param)  
}
```

- ① Sachant que  $\mathbb{E}(X) = \frac{a}{2}$ , créer une fonction `estim` qui prend en argument un échantillon de taille quelconque issu de  $X$  et renvoie l'estimation, par la méthode des moments, du paramètre  $a$ .
- ② Lancer plusieurs fois la fonction précédente avec un échantillon de taille 100. Quel semble être le paramètre  $a$  ?

Au lieu d'exécuter manuellement plusieurs fois l'instruction `estim(runifa(n))`, on va utiliser une fonction R qui le fait à notre place et accumule les différents résultats : il s'agit de la fonction `replicate` qu'on utilise comme suit :

```
a <- replicate(1000, estim(runifa(n)))
```

Le vecteur `a` stocke les 1000 résultats de l'instruction `estim(runifa(n))`.

- ③ Faire un diagramme en boîte de 1000 estimations successives de `a`. Quel est le paramètre inconnu ? Vérifier qu'il est en accord avec le vrai paramètre choisi au hasard, utilisé pour générer les observations et stocké dans `param`.



- ④ On admet que les moments de  $X$  sont :

$$m_k = \mathbb{E}(X^k) = \frac{a^k}{k+1} \quad \text{pour tout } k \geq 1.$$

Trouver pour chaque moment l'estimateur  $\hat{a}_k$  correspondant. Refaire l'étude précédente. Quel est l'estimateur le plus précis ?

## 2 Théorème central limite

On souhaite vérifier expérimentalement le théorème central limite. Pour cela, on va choisir une variable aléatoire  $X$  de loi quelconque, d'espérance  $\mu$  et d'écart-type  $\sigma$ . On utilisera la fonction suivante pour générer un échantillon de loi inconnue de taille  $n$  :

```
runknown <- function(n) {
  bn <- rbinom(n, 1, 0.2)
  bn * rnorm(n, mean=-4, sd=1) + (1 - bn) * rnorm(n, mean=10, sd=1)
}
```

- ⑤ Vérifier expérimentalement que l'espérance de  $X$  vaut  $\mu = 7.2$  et que l'écart-type vaut  $\sigma = \sqrt{32.36}$ .
- ⑥ Tracer un histogramme d'un échantillon de taille 1000.
- ⑦ Tracer la fonction de répartition empirique de la loi inconnue avec un échantillon de taille 1000 à l'aide des fonctions `ecdf` et `plot`. Commenter la cohérence du graphe de la fonction de répartition avec celui de l'histogramme.

Le théorème central limite dit que si on a  $n$  variables aléatoires iid  $X_1, \dots, X_n$  de v.a. parente  $X$ , alors la variable aléatoire suivante,

$$T = \frac{\bar{X} - \mu}{\sigma/\sqrt{n}},$$

converge en loi lorsque  $n$  augmente vers une loi normale centrée réduite. On va donc vérifier que des réalisations issues de la loi de  $T$  sont distribuées comme une gaussienne centrée réduite.

- ⑧ À partir de l'échantillon de taille  $n$  de loi celle de  $X$ , calculer une seule réalisation de la variable aléatoire  $T$ .



- ⑨ Répéter l'opération précédente plusieurs fois en régénérant à chaque fois un nouvel échantillon de taille  $n$  de  $X$ . À quoi semble ressembler la distribution obtenue ?

Pour générer un nombre quelconque de réalisations de  $T$ , on va à nouveau utiliser la fonction `replicate`. Il faut d'abord créer une fonction qui regroupe le calcul de la réalisation et retourne le résultat.

```

random.T <- function(n) {
  # Générer le vecteur x de taille n
  # Calculer une réalisation de la loi T
  return(valeur)
}

```

Pour avoir une réalisation de la loi  $T$ , il suffit maintenant d'appeler la fonction `random.T` comme ceci

```
| random.T(n)
```

Ensuite, pour appeler `random.T` un nombre quelconque de fois et stocker les différents résultats dans un vecteur, on exécute

```
| t.10 <- replicate(10, random.T(n))
```

qui stocke dans le vecteur `t.10` 10 appels successifs de la fonction `random.T` avec l'argument  $n$ .

- ⑩ Définir la fonction `random.T` et générer un vecteur `t.1000` contenant 1000 réalisations de la variable aléatoire  $T$ . Vérifier que la moyenne empirique et la variance empirique sont en accord avec une loi normale centrée réduite.
- ⑪ Tracer la fonction de répartition empirique de l'échantillon `t.1000`; comparer la avec celle d'un échantillon `x` vu à la question 7.
- ⑫ Avec l'instruction `curve(pnorm, add=TRUE)`, superposer au graphe précédent, la fonction de répartition théorique d'une gaussienne centrée réduite. Qu'observez-vous?
- ⑬ En reprenant la question précédente, illustrer graphiquement le théorème central limite.

### 3 Estimation par maximum de vraisemblance

On considère la loi exponentielle de paramètre  $\lambda > 0$  et de densité

$$f_{\lambda}(x) = \begin{cases} \lambda e^{-\lambda x} & \text{si } x \geq 0 \\ 0 & \text{sinon,} \end{cases}$$

disponible en R avec la fonction `dexp`. On fixe le paramètre  $\lambda = 3$  et on suppose qu'on dispose d'un échantillon iid  $x_1, \dots, x_n$  issu de cette loi. On cherche à présent à estimer le paramètre  $\lambda$  qu'on suppose à présent inconnu avec la méthode du maximum de vraisemblance; on note  $\hat{\lambda}$  l'estimateur du maximum de vraisemblance de  $\lambda$  et  $\hat{\lambda}^{real}$  une de ses réalisations.

- ⑭ Créer la fonction `f` de densité qui prend en argument un paramètre  $\lambda$  et un échantillon  $x$  et renvoie les densités aux points `x` pour la loi exponentielle de paramètre  $\lambda$ .



- (15) Créer la fonction `L` de vraisemblance qui prend en argument un paramètre  $\lambda$  et un échantillon  $x$  et renvoie la vraisemblance du paramètre  $\lambda$  par rapport aux données  $x$ . On pourra utiliser la fonction `prod`.
- (16) Créer la fonction `logL` de log-vraisemblance qui prend en argument un paramètre  $\lambda$  et un échantillon  $x$  et renvoie la log-vraisemblance du paramètre  $\lambda$  par rapport aux données  $x$ . Pour des raisons de stabilité, on utilisera le fait que le logarithme d'un produit est la somme des logarithmes<sup>1</sup>.
- (17) Stocker dans la variable `x` un échantillon de taille  $n = 100$  suivant une loi exponentielle de paramètre  $\lambda = 3$ . D'après la vraisemblance, quelle est la valeur du paramètre la plus probable entre  $\lambda = 3.1$  et  $\lambda = 2.8$ ?
- (18) Que fait le bout de code suivant

```

| lambdas <- seq(0, 6, 0.01)
| logL.lambdas <- sapply(lambdas, function(lambda) logL(lambda, x))
| plot(lambdas, logL.lambdas, type = "l")

```

À partir d'un échantillon `x`, la méthode du maximum de vraisemblance consiste à trouver la valeur  $\hat{\lambda}^{real}$  du paramètre  $\lambda$  qui maximise la fonction `logL`. Pour ce faire, nous allons utiliser la fonction `optimize` de R. Supposons qu'on souhaite calculer le maximum de la fonction  $g(x) = -(x - \pi)^2$  (qui vaut 0 et est atteint pour  $x = \pi$ ). On définit d'abord la fonction `g` en R :

```

| g <- function(x) {
|   -(x - pi)^2
| }

```

On appelle ensuite la fonction `optimize` comme suit :

```

| (opt <- optimize(g, lower = -10, upper = 10, maximum = TRUE))
| $maximum
| [1] 3.141593
| $objective
| [1] 0

```

Les arguments `upper` et `lower` fixe l'intervalle de recherche et `maximum` spécifie que l'on recherche un maximum et non un minimum.

La fonction retourne un objet que l'on a pas encore rencontré : une liste nommée. Il s'agit d'une liste contenant des objets qui sont accessibles en fournissant leur nom.

```

| opt$maximum
| [1] 3.141593
| opt$objective

```

1. Comparer `log(1e-200 * 1e-200)` et `log(1e-200) + log(1e-200)`

```
| [1] 0
```

Attention, si on veut optimiser une fonction à plusieurs arguments (par exemple une fonction de vraisemblance), il ne faut pas oublier de fixer dans la fonction `optimize` les arguments sur lesquels ne porte pas l'optimisation. À titre d'exemple, on pourra consulter la section « Exemples » de l'aide sur la fonction `optimize`.

- 19 Calculer la valeur  $\hat{\lambda}^{real}$  du paramètre  $\lambda$  la plus vraisemblable par rapport à l'échantillon  $\mathbf{x}$ .

De la même manière que dans la section précédente, on cherche maintenant à recommencer cette estimation pour différents échantillons  $\mathbf{x}$ .

- 20 Créer une fonction `sim.EMV` qui ne prend aucun argument, définit un échantillon et calcule  $\hat{\lambda}^{real}$ , la réalisation du maximum de vraisemblance associée à cet échantillon.

- 21 À partir de 10000 simulations de  $\hat{\lambda}^{real}$  (on utilisera pour cela les fonctions `replicate` et `sim.EMV`) et de la méthode des moments, déterminer une estimation de  $\mathbb{E}(\hat{\lambda})$  et de  $\text{Var}(\hat{\lambda})$  et comparer l'estimation du biais de  $\hat{\lambda}$  avec son biais théorique dont on admet qu'il vaut

$$\frac{n}{n-1}\lambda - \lambda.$$



## 4 Information de Fisher

Nous allons à présent calculer une valeur approchée de l'information de Fisher pour le paramètre  $\lambda = 3$ . Pour calculer l'information de Fisher, il faut pouvoir calculer la dérivée de la log-vraisemblance au point  $\lambda = 3$ . Pour ce faire, nous allons faire appel à une fonction appartenant à une bibliothèque qui n'est pas installée par défaut. Pour installer une bibliothèque il suffit d'exécuter l'instruction

```
| install.packages("ma-bibliothèque")
```

puis, pour charger la bibliothèque, on exécute

```
| library(ma-bibliothèque)
```

- 22 Installer puis charger la bibliothèque `pracma`.

Nous allons utiliser la fonction `grad` de la bibliothèque `pracma`. Elle calcule la dérivée d'une fonction en un point. Si on reprend l'exemple précédent avec la fonction  $g$  :

```
| grad(g, 0)
| [1] 6.283185
```

On trouve bien  $g'(0) = 2\pi$ .

- 23 Créer une fonction `sim.Fisher` sans argument qui génère un échantillon  $\mathbf{x}$ , et calcule l'information de Fisher de cet échantillon pour  $\lambda = 3$ .

②④ Encore une fois, utiliser `sim.Fisher` et `replicate` pour simuler 1000 fois le calcul de l'information de Fisher d'un échantillon et donner une estimation de l'information de Fisher que l'on comparera à sa valeur théorique donnée par,

$$I_n(\lambda) = \frac{n}{\lambda^2}.$$



②⑤ Sachant que l'estimateur du maximum de vraisemblance est asymptotiquement normal, estimer sa variance à l'aide de l'information de Fisher. Comparer le résultat obtenu avec la variance empirique de l'estimateur.



②⑥ Retrouver expérimentalement le fait que,

$$I_n(\lambda) = -\mathbb{E} \left[ \frac{\partial^2 \log L}{\partial \lambda^2}(\lambda; X_1, \dots, X_n) \right].$$

On pourra d'abord définir la fonction `grad2` prenant en argument une fonction `f` et un point `x` et qui calcule la dérivée seconde de la fonction `f` au point `x`.