

# Introduction à bootstrap

M2R santé publique

03 octobre 2017

# Principe

- ▶ On s'intéresse au paramètre d'une loi de probabilité  $F_\theta$ ,  $f_\theta$ ,  $f(x; \theta)$  ou  $\mathbb{P}_\theta$
- ▶ On ne possède qu'un **échantillon**  $X_1, \dots, X_n$  issu de  $F_\theta$
- ▶  $\hat{\theta} = t(X_1, \dots, X_n)$  est un estimateur de  $\theta$

# Principe

- ▶ On s'intéresse au paramètre d'une loi de probabilité  $F_\theta$ ,  $f_\theta$ ,  $f(x; \theta)$  ou  $\mathbb{P}_\theta$
- ▶ On ne possède qu'un **échantillon**  $X_1, \dots, X_n$  issu de  $F_\theta$
- ▶  $\hat{\theta} = t(X_1, \dots, X_n)$  est un estimateur de  $\theta$

Plusieurs **questions** liées à la distribution de  $\hat{\theta}$

- ▶ **Biais** de l'estimateur  $\hat{\theta}$
- ▶ **Variance** de  $\hat{\theta}$  (sa précision)
- ▶ **Intervalle de confiance** pour  $\theta$

## Simuler suivant $F_\theta$ ou $f_\theta$

Exemple : supposons que  $X \sim \mathcal{N}(\theta, 1)$  où  $\theta = 2$

```
set.seed(123)
# simuler un jeu de données de 100 individus
n <- 100
x <- rnorm(n, mean=2, sd=1)

# simuler N jeux de données
N <- 10000
X <- matrix(rnorm(N*n, 2, 1), N, n)
t <- rowMeans(X)
summary(t)
```

## Le résultat

```
set.seed(123)
# simuler un jeu de données de 100 individus
n <- 100
x <- rnorm(n, mean=2, sd=1)

# simuler N jeux de données
N <- 10000
X <- matrix(rnorm(N*n,2,1), N,n)
t <- rowMeans(X)
summary(t)
```

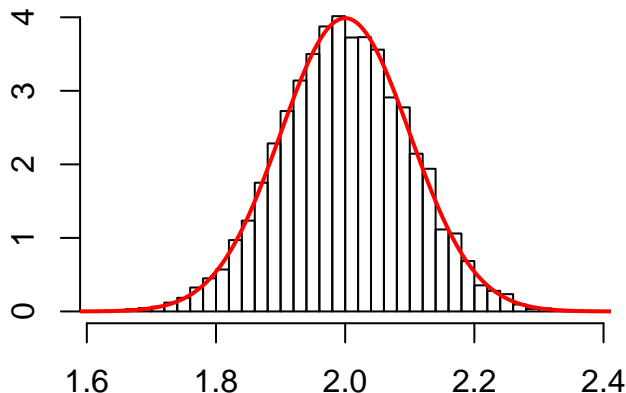
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1.623	1.932	1.999	1.999	2.068	2.376

## Le code pour tracer les sorties

```
hist(t, nclass=40, freq=F, ylab="",xlab="",main="")
xx <- seq(1.5, 2.5, le=200)
et <- 1/sqrt(n)
points(xx, dnorm(xx, 2, et),
       type="l", lwd = 2, col="red")
```

## La figure

```
hist(t, nclass=40, freq=F, ylab="", xlab="", main="")  
xx <- seq(1.5, 2.5, le=200)  
et <- 1/sqrt(n)  
points(xx, dnorm(xx, 2, et), type="l", lwd = 2, col="red")
```



## Une petite parenthèse : générateur de nombres aléatoires

La loi de base que l'on simule est la loi uniforme sur  $[0, 1]$ , un tel générateur est inclus dans tous les langages de programmation et dans tous les logiciels.



## Une petite parenthèse : générateur de nombres aléatoires

La loi de base que l'on simule est la loi uniforme sur  $[0, 1]$ , un tel générateur est inclus dans tous les langages de programmation et dans tous les logiciels.

- ▶ La méthode de congruence linéaire (D.H Lehmer-Lucas 1948),

$$g_{n+1} = ag_n + b \mod m$$

où  $a, b, m$  et  $g_0$  sont à choisir initialement.

- ▶ Ce générateur fabrique une suite d'entiers aléatoires allant de 0 à  $m - 1$ .
- ▶ La suite  $g_n/m - 1$  fournit des nombres sur l'intervalle  $[0, 1]$ .
- ▶ La suite ainsi produite possède une période inférieure ou égale à  $m$ .
- ▶ Un générateur de ce type, nommé RANDU, fut exploité pendant des années dans toutes sortes de logiciels.
- ▶ Les paramètres de RANDU sont  $m = 2^{35}$ ,  $a = 3125$ ,  $b = 1$  et  $g_0 = 71$  et sa période est de  $2^{35}$

# Inversion de la fonction de répartition

On veut générer des nombres aléatoires suivant une loi données par sa fonction de répartition  $F$ . Comment procéder ?

# Inversion de la fonction de répartition

On veut générer des nombres aléatoires suivant une loi données par sa fonction de répartition  $F$ . Comment procéder ?

- ▶ Générer  $U_1, U_2, \dots, U_n$  suivant une loi uniforme  $\mathcal{U}(0, 1)$ .
- ▶ On forme

$$(X_1, X_2, \dots, X_n) = (F^{-1}(U_1), F^{-1}(U_2), \dots, F^{-1}(U_n))$$

# Inversion de la fonction de répartition

On va s'intéresser à la loi exponentielle  $\mathcal{E}(\lambda)$  où  $\lambda = 2$

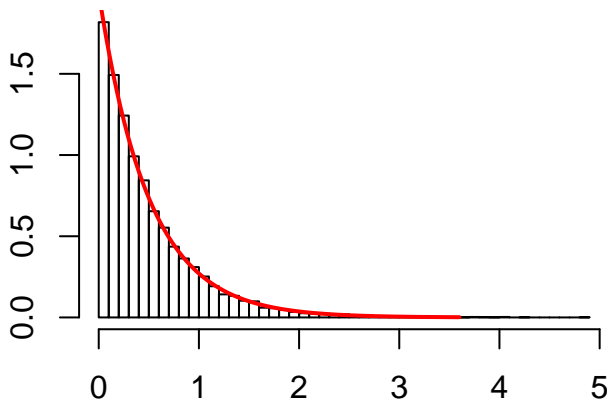
- ▶ Rappelons que  $F(x) = 1 - e^{-\lambda x}$
- ▶ Simuler 100 réalisations de la loi  $\mathcal{E}(2)$
- ▶ Tracer sur la même figure l'histogramme des réalisations ainsi que la vraie densité de probabilité de la loi  $\mathcal{E}(2)$

## Inversion de la fonction de répartition

```
set.seed(123)
x.u <- runif(1000)
x.e <- qexp(x.u, rate = 2)
hist(x.e, nclass=30, freq = F, main="", xlab="", xlab="")
x<-seq(0,3.6, le=100)
points(x, dexp(x, 2), type="l", lwd = 2, col = "red")
```

## Sortie

```
set.seed(123)
x.u <- runif(10000)
x.e <- qexp(x.u, rate = 2)
hist(x.e, nclass=40, freq = F, main="", ylab="", xlab="")
x<-seq(0,3.6, le=100)
points(x, dexp(x, 2), type="l", lwd = 2, col = "red")
```



## Simuler une loi normale : **Box-Muller**

Soit,  $U$  et  $V$  deux variables aléatoires **uniformes** sur  $[0, 1]$  indépendantes, la variable

$$X = \cos(2\pi U) \sqrt{(-2 \log(V))}$$

suit une loi **normale**  $\mathcal{N}(0, 1)$ .

## Alternative : rééchantillonnage

$F_\theta$  est en général inconnue !



## Alternative : rééchantillonnage

$F_\theta$  est en général inconnue !

Le **bootstrap** consiste donc à faire une simulation à partir de la loi empirique  $F_n$  observée (i.e. l'échantillon) au lieu de la vraie loi  $F_\theta$ , qui est inconnue.

## Alternative : rééchantillonnage

$F_\theta$  est en général inconnue !

Le **bootstrap** consiste donc à faire une simulation à partir de la loi empirique  $F_n$  observée (i.e. l'échantillon) au lieu de la vraie loi  $F_\theta$ , qui est inconnue.

- ▶ On tire  $B$  échantillons  $\mathbf{x}_1^*, \mathbf{x}_2^*, \dots, \mathbf{x}_B^*$  où chaque  $\mathbf{x}_b^*$  est constitué en tirant avec remise  $n$  valeurs de l'échantillon  $X_1, \dots, X_n$ .
- ▶ On appelle chaque tirage d'un  $\mathbf{x}_b^*$  un **bootstrap**

# La fonction sample de R

- Sous R, taper :

```
?sample
```

- On propose de faire un premier test avec cette fonction

```
set.seed(123)
# 100 réalisations d'une gaussienne N(10,10)
x <- rnorm(100, mean = 10, sd = 10)
# faire un tirage avec remise de taille 100
x.boot <- sample(?, ?, ?) ## attention remplacer les ?
```

# Un jeu de données

**Exemple :** jeu de données de taille  $n = 47$  observations de la durée de rémission d'une leucémie en semaines.

```
## lecture du jeu de données leucemie
setwd("~/Dropbox/enseignements/bootstrap")
rm(list = ls())
## je vais lire le fichier leucemie.txt
load("~/Dropbox/enseignements/bootstrap/leucemie.rda")
```

## Exemple : estimation de la médiane

- ▶ On s'intéresse à la médiane dans le jeu de données de leucémie, ici la statistique  $t(\mathbf{x})$  est la médiane empirique de l'échantillon  $\mathbf{x} = (X_1, \dots, X_n)$ .
- ▶ Proposer une procédure qui génère  $B = 1000$  échantillons bootstrap à partir du jeu de données leucémie.
- ▶ Calculer l'estimateur bootstrap du biais donné par

$$\widehat{\text{biais}} = \frac{1}{B} \sum_{b=1}^B \text{median}(\mathbf{x}_b^*) - \text{median}(X_1, \dots, X_n)$$

- ▶ Estimer la variance et l'écart type de l'estimateur de la médiane.

# Le package boot de R

On souhaite ici, voir ce que fait le package boot et le comparer à nos résultats sur le jeux de données leucémie.

```
set.seed(123)
boot.median<-function(y,i){z<-median(y[i]); return(z)}
boot.obj1<-boot(data = y,
                statistic = boot.median,
                R = B)
names(boot.obj1)
boot.obj1
```

# Bilan

- ▶ La variabilité de l'échantillonnage dans la population se reflète dans la variabilité de l'échantillonnage dans la sous-population  $y$ .
- ▶ Nous avons utilisé la variance du vecteur *boot* pour estimer la variance de *median*( $y$ ).
- ▶ Il est clair que ce que nous venons de faire pour la médiane, nous pouvons le répéter pour estimer la variance d'autres statistiques.

# Intervalles de confiance : jeu de données CommuteAtlanta

- ▶ Charger le package Lock5Data et le jeu de données CommuteAtlanta
- ▶ Se faire une idée du jeu de données : sa taille, les noms des variables etc ...
- ▶ Générer  $B = 1000$  échantillons bootstrap sans faire appel au package boot de la variable *trajet* (Distance)
- ▶ Calculer l'écart type du trajet moyen
- ▶ Construire un intervalle de confiance du trajet moyen