**EILCO** **2023-2024**
**Cycle Ingénieur 1ère année**

**TP3 : Lois de probabilités, simulations, optimisation**

# 1 Lois de probabilités

Pour une variable aléatoire X de loi *law* :

— `dlaw(x, paramètres)` calcule $\mathbb{P}(X = x)$ si $X$ est discrète et $f_X(x)$ si $X$ est continue de densité $f_X$ ;

— `plaw(q,paramètres)` calcule la fonction de répartition ;

— `qlaw(p, paramètres)` donne le quantile d'ordre $p$ ;

— `rlaw(m,parametres)` génère $m$ nombres aléatoires selon la loi, cad un échantillon i.i.d. de taille $m$ de $X$.

La fonction `set.seed()` permet de spécifier l'amorce du générateur de nombre aléatoire, ce qui se révèle utile quand on veut répéter une simulation à l'identique (les deux quantités générées ci-dessous sont les mêmes) :

```
set.seed(42); runif(1)
set.seed(42); runif(1)
```

## 1.1 Lois classiques

`R` sait générer un grand nombre de lois usuelles. Les plus classiques sont les suivantes :

**Lois discrètes**

— loi uniforme $\mathscr{U}(\{1,\ldots,n\})$ : `sample(1:n,m,replace=T)` ;

— loi de Bernoulli $\mathscr{B}(p)$ : `rbinom(m,size=1,prob=p)` ;

— loi binomiale $\mathscr{B}(n,p)$ : `rbinom(m,size=n,prob=p)` ;

— loi géométrique $\mathscr{G}(p)$ : `rgeom(m, prob=p)` ;

— loi de Poisson $\mathscr{P}(\lambda)$ : `rpois(m,lambda=lambda)`.

**Lois continues**

— loi uniforme $\mathcal{U}[a,b]$ : `runif(m,min=a,max=b)` ;

— loi normale $\mathcal{N}(\mu,\sigma^2)$ : `rnorm(m,mean=mu,sd=sigma)` ;

— loi exponentielle $\mathcal{E}(\lambda)$ : `rexp(m,rate=lambda)` ;

— loi du Chi-deux à $r$ degrés de liberté $\chi^2(r)$ : `rchisq(m,df=r)` ;

— loi de Student à $r$ degrés de liberté : `rt(m,df=r)`.

## 1.2 Loi multinomiale

On utilise :

```
sample(x=urne,size=nombre boules à extraire,replace=TRUE/FALSE, prob
= probabilité des differents couleurs)
```

**Exemple 1.** On tire 1000 fois une pièce de monnaie biaisée telle que la probabilité d'obtenir face est 0.3 et on calcule ensuite la proportion des faces obtenues.

```
echantillon<-sample(x=c(0,1), size=1000, replace=T, prob=c(0.7, 0.3))
plot(cumsum(echantillon)/(1:1000), main="Loi des grands nombres", type="l",
ylim=c(0,0.7), xlab="Taille échantillon")
abline(h=0.3,col=2)
legend(x="top", lty=1, col=1:2, legend=c("fréquence empirique de face",
"probabilité d'avoir face"))
```

Noter qu'à la place d'utiliser `sample()` on aurait pu utiliser `rbinom(n=1000, size=1, prob=0.3)`.

**Exemple 2.** Cinq candidats se présentent à une élection. On effectue un sondage auprès de 1000 personnes et on calcule la proportion de bulletins récoltés par chaque candidat.

```
candidats<-letters[1:5]
echant<-sample(x=candidats, size=1000,replace=TRUE,prob=c(.2,.16,.26,.25,.13))
barplot(summary(as.factor(echant)), ylab="Votes", xlab="Candidats", main="Echant.
électeurs")
```

# 2 La famille `_apply()`

Dans de nombreux cas, il est possible de remplacer une boucle `for` par un appel beaucoup plus efficace (en terme de temps d'exécution et de mémoire utilisée) à une fonction de type `_apply()` ou `replicate()`.

### 2.1 La fonction `apply()`

Taper les lignes suivantes :

```
M<-matrix(1:9,3,3)
apply(M,1,sum); apply(M, 1, var); apply(M, 2, min)
x <- array(sample(1:10, 36, rep = TRUE), c(3, 3, 4))
apply(x,3, det)
apply(x,2, det) #n'a pas de sens
apply(x, c(1,2), sum)
apply(x,c(2,3),sum)
```

### 2.2 Les fonctions `lapply` et `sapply`

La fonction `apply` s'applique à des matrices. Les fonctions `lapply` et `sapply` sont plus générales au sens où elles s'appliquent à des listes. Taper, par exemple :

```
(x<-list(runif(7), runif(5)))
lapply(X=x,FUN=mean)
lapply(X=c(3,10), FUN=sample, x=1:10, replace=T)
sapply(X=c(3,3),sample, x=1:10,replace=T)
```

### 2.3 La fonction `replicate()`

Pour exécuter *n* fois une expression *expr*, on utilise la fonction enveloppante de `sapply()` : `replicate(n,expr)`. Cette fonction est très utile quand on fait des simulations. Taper par exemple :

```
sapply(rep(1, 10), function(i) sample(1:100, 12))
replicate(10, sample(1:100, 12))
```

Les deux lignes ci-dessus sont équivalentes, mais la seconde est plus élégante.

## 3 Optimisation du temps d'exécution

### 3.1 Mesure du temps d'exécution

La fonction `proc.time()` donne le temps écoulé à partir du début de la session `R`. Pour mesurer le temps d'exécution, utiliser la fonction `system.time(expr)` :

```
system.time(for ( i in 1:10000)print(i))
system.time(for ( i in 1:10000)cat(i))
```

### 3.2 Comparaisons de différentes méthodes dans un exemple

On crée un data frame avec quatre colonnes numériques :

```
col1<-runif(10000, 0, 2)
col2<-runif(10000, 0, 2)
col3<-rnorm(10000, 0, 1)
col4<-rnorm(10000, 0, 1)
df<-data.frame(col1, col2, col3, col4)
head(df)
```

On veut ajouter une cinquième colonne avec les étiquettes `sup_a_trois` ou `inf_a_trois` si la somme des quatre valeurs sur une même ligne est supérieure ou inférieure à trois.

### Approche par boucle

```
s1 <- system.time({
for (i in 1:10000){
if (df[i,1]+df[i,2]+df[i,3]+df[i,4]>3) df[i,5]<-"sup_a_trois"
else df[i,5]<-"inf_a_trois"
}
})
s1
```

### Pré-allocation

Pour ne pas avoir à incrémenter la taille de df à chaque itération dans la boucle, ce qui est peu efficace, il convient d'initialiser les variables de sortie avant la boucle :

```
s2 <- system.time({
output<-character(length=10000)
for (i in 1:10000){
if (df[i,1]+df[i,2]+df[i,3]+df[i,4]>3) output[i]<-"sup_a_trois"
else output[i]<-"inf_a_trois"
}
df$output=output
})
s2
```

### Pré-allocation `ifelse()`

La fonction `ifelse()` permet de faire du traitement conditionnel de façon plus efficace que `if(){} else{}` :

```
s3 <- system.time({
output<-character(length=10000)
for (i in 1:10000){
```

```
output <- ifelse (df[i,1]+df[i,2]+df[i,3]+df[i,4]>3, "sup_a_trois", "inf_a_trois")
}
})
s3
```

**En utilisant les fonctions** `apply()`

```
ma_fonction<-function(x){
ifelse (x[1]+x[2]+x[3]+x[4]>3, "sup_a_trois", "inf_a_trois")
}
s4 <- system.time({
df$output<-apply(df[,1:4], 1, ma_fonction)
})
s4
```

# 4 Exercices

**Exercice 1** On rappelle qu'une variable aléatoire $Y$ suit la *loi log-normale* s'il existe une variable aléatoire $X$ suivant la loi normale de paramètre $(\mu, \sigma^2)$ telle que $Y = e^X$. Simuler 1 000 réalisations d'une loi log-normale de paramètres $\mu = \log(5000) - \frac{1}{2}$ et $\sigma^2 = 1$, puis tracer l'histogramme de l'échantillon aléatoire obtenu.

**Exercice 2**

1. (a) Générer 1000 réalisations d'une loi binomiale de paramètres $n = 8$ et $p = 0.25$.
   (b) Donner la proportion de nombre de fois que l'on a obtenu 1 à partir de l'échantillon obtenu.
2. Déterminer les probabilités suivantes :
   (a) probabilité d'obtenir 1 avec une loi binomiale $\mathscr{B}(8, 0.25)$ ;
   (b) probabilité d'obtenir plus que 2 et moins que 7 (au sens large) avec une loi binomiale $\mathscr{B}(8, 0.25)$ ;
   (c) probabilité d'obtenir au plus 1 avec une loi binomiale $\mathscr{B}(8, 0.25)$ ;
   (d) probabilité d'obtenir plus que 3 (au sens large) pour une loi de Poisson de paramètre $\lambda = 4$ ;
   (e) probabilité d'obtenir plus que 2. 07 pour une loi normale centrée réduite.
3. (a) Quelle est la valeur $x$ telle que $\mathbb{P}(X \leq x) = 0.95$ pour une loi normale centrée réduite ?

(b) Quel est le quantile 1% pour une loi de Student à 5 degrés de liberté ?

**Exercice 3**

1. Tracer les densités de la loi uniforme de paramètres $a = -1$ et $b = 1$ et de la loi normale de paramètre $(\mu, \sigma^2) = (0, 4)$.
2. Simuler $n = 1000$ réalisations indépendantes de variables aléatoires suivant la loi de Bernoulli de paramètre 0.5, puis la loi normale de paramètre $(\mu, \sigma^2) = (0, 4)$. Calculer les moyennes et les variances empiriques de ces échantillons et les comparer à leurs valeurs théoriques.
3. Taper le code ci-dessous.

```
n <- 20
X <- runif(n, min=-1, max=1)
nb <- 1000
t <- seq(-1, 1, length.out=nb)
hist(X, freq=FALSE)
f <- dunif(t, min=-1, max=1)
lines(t, f, col="red")
```

Faire varier la valeur de $n$. Que constate-t-on ?

**Exercice 4**

1. Pour $n$ variant de 1 à 1000, simuler des échantillons indépendants de loi uniforme sur $[0, 1]$. Pour chaque échantillon de taille $n$, calculer la moyenne empirique $\overline{x}_n$. Tracer l'évolution de cette statistique en fonction de $n$. Vers quelle valeur converge la suite $\overline{x}_n$ ?
2. Etudier la convergence de la variance empirique.

**Exercice 5** Vérifier que la loi binomiale $\mathscr{B}(10000, 0.0005)$ peut être approchée par la loi de Poisson $\mathscr{P}(5)$.

**Exercice 6**

1. Simuler un échantillon, de taille $N = 10000$, de loi exponentielle de paramètre $\lambda = 2$ et tracer l'histogramme.
2. Illustrer la loi des grands nombres.
3. Illustrer le théorème centrale limite en simulant $M = 5000$ échantillons de taille $N = 10000$.

**Exercice 7**

1. Soient $f$ et $g$ des fonctions continues sur un intervalle $I$. On suppose que $f$ soit une densité de probabilité. Justifier que si $X_1, \ldots, X_n$ est une suite de variables aléatoires i.i.d. de densité $f$ telles que $g(X_1)$ admette une espérance, alors
$$\frac{1}{n}\sum_{i=1}^{n} g(X_i) \underset{n\to\infty}{\longrightarrow} \int_I g(x)f(x)\mathrm{d}x \quad \text{p. s.}$$
2. En simulant 10000 variables aléatoires suivant la loi uniforme dans $[0,1]$, donner une valeur approchée de $\int_0^1 \frac{\log(1+x)}{\sqrt{1+x^2}}\,\mathrm{d}x$. Comparer la valeur obtenue à `integrate(function(x) log(1+x)/sqrt(1+x^2), 0, 1)`.
3. Donner deux valeurs approchées de $\int_0^1 \sqrt{1-x^2}\,\mathrm{d}x$ en simulant des variables aléatoires dans le segment $[0,1]$, puis dans le carré $[0,1]^2$.

**Exercice 8** On cherche à avoir une estimation de $I = \int_0^\infty \exp(-t^2/2)\mathrm{d}t$.

1. Montrer que $I = \sqrt{\frac{\pi}{2}}$.
2. En effectuant le changement de variables $x = t^2/2$, prouver que
$$I = \int_0^\infty \frac{1}{\sqrt{2x}}\exp(-x)\mathrm{d}x.$$
3. Montrer que cette intégrale peut être vue comme l'espérance d'une fonction d'une variable aléatoire $X$ qui suit une loi exponentielle. Donner le paramètre de cette loi exponentielle.
4. Mettre en oeuvre un programme permettant d'estimer cette intégrale. Comparer avec la fonction `integrate`.

**Exercice 9** La densité de probabilité et la fonction de répartition de la loi de Pareto de paramètres $\alpha$ et $\lambda$, sont respectivement
$$f(x) = \begin{cases} \dfrac{\alpha\lambda^\alpha}{(x+\lambda)^{\alpha+1}} & \text{si } x \geq 0 \\ 0 & \text{sinon.} \end{cases}$$
et
$$F(x) = \begin{cases} 1 - \left(\dfrac{\lambda}{x+\lambda}\right)^\alpha & \text{si } x \geq 0; \\ 0 & \text{sinon.} \end{cases}$$
Le but de cet exercice est de simuler la loi de Pareto.

1. Tracer le graphe de la densité lorsque $\alpha = 200$ et $\lambda = 10$.
2. On désigne par $F^{-1}$ la fonction de répartition (pseudo-) inverse de $F$. Montrer que, pour tout $u \in ]0,1[$, on a

$$F^{-1}(u) = \lambda\left((1-u)^{-1/\alpha} - 1\right).$$

3. Montrer que si $U$ suit la loi uniforme sur $[0,1]$ alors $F^{-1}(U)$ suit la loi de Pareto de paramètres $\alpha$ et $\lambda$.
4. Simuler $n = 10000$ valeurs i.i.d. selon une loi de Pareto de paramètres $\alpha = 200$ et $\lambda = 10$.
5. Tracer l'histogramme en superposant la densité de la loi de Pareto.