Statistiques Inférentielles II
Correction du TD 2 : Simulations

Licence 3 MIASHS - IDS (2021-2022)

Jairo Cuglirari, Guillaume Metzler
Institut de Communication (ICOM)
Université de Lyon, Université Lumière Lyon 2
Laboratoire ERIC UR 3083, Lyon, France
jairo.cugliari@univ-lyon2.fr guillaume.metzler@univ-lyon2.fr

Exercice 1 : Ecriture binaire

1. Ecrire 45 en base 2.

   On rappelle que pour obtenir l'écriture binaire d'un nombre, il suffit de faire une succession de divisions euclidiennes par 2 et de réécrire la suite des restes.

$$\begin{aligned} 45 &= 2 \times 22 + 1, \\ 22 &= 2 \times 11 + 0, \\ 11 &= 2 \times 5 + 1, \\ 5 &= 2 \times 2 + 1, \\ 2 &= 2 \times 1 + 0, \\ 1 &= 2 \times 0 + 1, \end{aligned}$$

   On remonte cette série de divisions eucldienne de bas en haut pour obtenir le résultat.

$$(45)_2 = 101101.$$

2. Faisons de même avec $45 \pmod{16}$ et $45 \pmod{17}$.

   Le reste de la division euclidienne de 45 par 16 est 13. Son écriture en base 2 est alors

   $$(13)_2 = 1101.$$

   Le reste de la division euclidienne de 45 par 17 est 11. Son écriture en base 2 est alors

   $$(11)_2 = 1011.$$

3. Que remarquez vous lorsque vous regardez ces trois représentations ?

   La première représentation est symétrique. On retrouve le schéma des représentations deux et trois dans la première représentation. Enfin, les représentations deux et trois sont quasi identiques, à une permutation près.

4. Ecrire un code permettant de générer la représentation binaire d'un nombre.

```
# Ecriture de la fonction
binary <- function(x){
    # on indique la base de la représentation
    base = 2
    # on créé un vecteur qui stocke les restes
    r = NULL
    # on va ensuite effectuer nos divisions euclidiennes

    while(x>0){
        r = c(r,x%%base)
        x = x%/%base
    }

    # on va ensuite mettre tout cela sous forme de "mot"
    # en rebroussant chemin à l'aide de la fonction rev.
    r = paste(rev(r), collapse = "")

    return(r)
}

# On peut ensuite tester notre fonction

binary(45)
```

```
## [1] "101101"
```

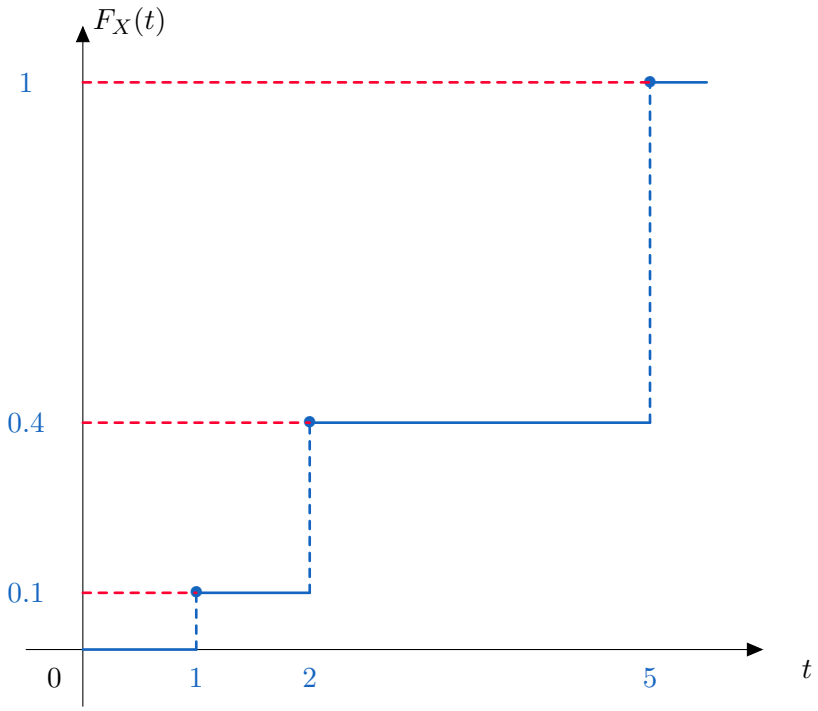
Exercice 2 : Simulation d'un échantillon

Considérons la variable aléatoire discrète avec la fonction de masse suivante :

$$\mathbb{P}[X = 1] = 0.1, \quad \mathbb{P}[X = 2] = 0.3, \quad \mathbb{P}[X = 5] = 0.6.$$

Représenter la fonction de répartition de cette variable aléatoire X et écrire un programme qui permet de générer un échantillon suivant cette distribution à l'aide de la loi uniforme.

On commence par le graphe de la fonction de répartition.



Ce que nous aurions pu faire avec le code suivant également

```
F <- function(x) {
  ifelse(x < 1, 0,
         ifelse(x <= 2, .1,
                ifelse(x <= 5, .4, 1)))
}
#curve(F, from = -3, to = 10, type = "s")
```

```
# décommenter pour afficher le graphe
```

Le code que nous devrons ainsi écrire se fera de la façon suivante, on commence par générer un nombre aléatoire u entre $[0, 1]$ selon une loi uniforme puis

$$X = \begin{cases} 1 & \text{si } u \in [0, 0.1], \\ 2 & \text{si } u \in]0.1, 0.4], \\ 5 & \text{si } u \in]0.4, 1]. \end{cases}$$

```
# Ecriture de la fonction
sample <- function(n,Fx,fx){
  # n : taille échantillon
  # fx: valeurs à affectées

  # stockage de notre échantillon
  x_ech = c()

  # on teste la valeur de u pour savoir
  # à quel intervalle elle appartient
  for (i in 1:n){
    u = runif(1)
    for (k in 1:length(Fx)){
      if (u > Fx[k]){
        x = fx[k]
      }
    }
    x_ech = c(x_ech,x)
  }
  return(x_ech)
}

Fx = c(0,0.1,0.4)
fx = c(1,2,5)

sample(10,Fx,fx)

##  [1] 1 2 1 5 5 5 5 2 5 2
```

Exercice 3 : Simulation d'une loi de Poisson

Soit X décrivant une loi de Poisson de paramètres λ, $\mathcal{P}(\lambda)$. On note $F(x) = \mathbb{P}[X \geq x]$ et $p(x) = \mathbb{P}[X = x]$.

1. Montrer que la fonction de masse vérifie

$$p(x+1) = \frac{\lambda}{x+1} p(x).$$

On rappelle qu'une variable aléatoire de poisson X admet la fonction de masse suivante pour tout $x \in \mathbb{N}$

$$\mathbb{P}[X = x] = \frac{\lambda^x}{x!} e^{-\lambda}.$$

Alors

$$\mathbb{P}[X = x+1] = \frac{\lambda^(x+1)}{(x+1)!} e^{-\lambda} = \frac{\lambda}{x+1} \frac{\lambda^x}{x!} e^{-\lambda} = \frac{\lambda}{x+1} \mathbb{P}[X = x].$$

2. A partir de cette relation, écrire une fonction permettant de calculer les valeurs des fonctions de masse et de répartition.

Pour calculer les valeurs de la fonction de répartition, il faudra simplement sommer les valeurs de la fonction de masse. On utilisera simplement le fait que $\mathbb{P}[X = x] = e^{-\lambda}$.

```r
compute <- function(x,lambda){

    # Initialisation des différentes quantités
    k=0
    p.x = exp(-lambda)
    F.x = p.x

    # Calcul
    while(k<x){
        p.x = lambda/(k+1)*p.x
        F.x = F.x + p.x
        k = k+1
    }

    # On retourne les résultats sous la forme d'une liste
    return(list(p.x = p.x, F.x = F.x))
```

```
}

# Test
compute(5,2)

## $p.x
## [1] 0.03608941
##
## $F.x
## [1] 0.9834364

# On peut comparer avec les fonctions suivantes
dpois(5,2)

## [1] 0.03608941

ppois(5,2)

## [1] 0.9834364
```

3. Si $X \in \mathbb{N}$ est une variable aléatoire et $F.x$ est une fonction qui retourne la valeur de F pour la variable aléatoire X, alors on peut simuler X avec le programme suivant

```
R.rand <- function(){
    u = runif(1)
    x = 0
    while(F(x) < u){
    x = x+1
    }
    return(x)
}
```

Dans le cas d'une distribution de Poisson, ce programme peut largement être améliorer en utilisant deux nouvelles variables $p.x$ et $F.x$.
Améliorer la fonction précédente en tenant compte de ces deux informations.

```
R.rand <- function(lambda){
    u = runif(1)
    x = 0
    p.x = exp(-lambda)
    F.x = p.x
    while(F.x < u){
    x = x+1
```

```
    p.x = (lambda/x)*p.x
    F.x = F.x + p.x
    }
    return(x)
}
```

Exercice 4 : Méthode de l'inverse I

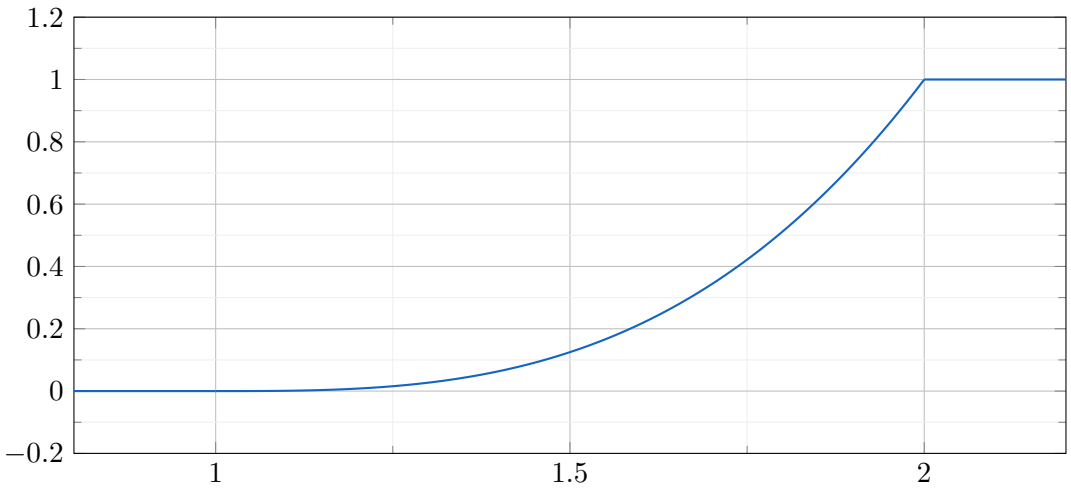
On considère la variable aléatoire continue dont la densité est donnée par

$$f_X(x) = \begin{cases} 3(x-1)^2 & \text{si } 1 < x \leq 2, 0 \\ \text{sinon.} \end{cases}$$

1. Tracer le graphe de la fonction de répartition associée.

La fonction de répartition est donnée, pour tout $t \in]1, 2]$ par

$$\begin{aligned} F(t) &= \mathbb{P}[X \leq t], \\ &= \int_1^t 3(x-1)^2 \, dx, \\ &= \frac{3}{3} \left[(x-1)^3\right]_1^t, \\ &= (t-1)^3. \end{aligned}$$



On aurait également pu faire cela avec R.

2. Montrer comment simuler une variable aléatoire ayant cette loi avec la méthode de l'inverse et écrire le code R correspondant.

   On commence par déterminer l'inverse de la fonction de répartition, notée Q, sur l'intervalle $]1, 2]$. Pour tout $p \in (0, 1)$, nous avons

   $$F(t) = p \iff t = F^{-1}(p) = Q(p).$$

   Cette fonction Q existe bien car F est bijective donc inversible. Alors Q est donnée par :

   $$\begin{aligned} F(t) &= p, \\ (t-1)^3 &= p, \\ t &= \sqrt[3]{p}, \\ t &= 1 + \sqrt[3]{p}. \end{aligned}$$

   On peut alors générer un échantillon de cette variable aléatoire avec le code suivant :

```
sample<-function(n){
    # Stockage de l'échantillon de taille n
    ech = c()
    for (i in 1:n){
        u = runif(1)
        ech = c(ech,1+u^(1/3))
    }
return(ech)
}

sample(3)
```

```
## [1] 1.957727 1.620240 1.882244
```

Exercice 5 : Méthode de l'inverse II

On considère la variable aléatoire continue X ayant la densité suivante pour tout réel x :

$$f_X(x) = \frac{\exp(-x)}{(1 + \exp(-x))^2}.$$

On dit que la variable aléatoire X a une distribution logistique (souvenez de la régression logistique qui a été présentée au semestre précédent).

1. Déterminer la fonction de répartition de cette variable aléatoire.

   Pour déterminer la fonction de répartition, on va avoir besoin de se rappeler que la dérivée de la fonction exponentielle est elle-même et que la dérivée de la fonction $u : x \mapsto u(x)^{-1}$ est la fonction $x \mapsto -\frac{u'(x)}{u(x)^2}$.

   $$\begin{aligned}
   F_X(t) &= \mathbb{P}[X \leq t], \\
   &\quad \downarrow \text{ en théorie il faudrait montrer que cette intégrale est bien définie} \\
   &= \int_{-\infty}^{t} \frac{\exp(-x)}{(1+\exp(-x))^2}\, dx, \\
   &\quad \downarrow \text{ on utilise l'indication pour chercher une primitive de la forme } (1+\exp(-x))^{-1} \\
   &= \left[\frac{1}{1+\exp(-x)}\right]_{-\infty}^{t}, \\
   &= \frac{1}{1+\exp(-t)}.
   \end{aligned}$$

2. Déterminer l'inverse de la fonction de répartition et écrire un code permettant de simuler un échantillon ayant cette loi à l'aide de la méthode de l'inverse.

   On procède comme à l'exercice précédent, on note que cette fonction est bijective sur $\mathbb{R}$ et est donc inversible.

   $$\begin{aligned}
   F(t) &= p, \\
   \frac{1}{1+\exp(-t)} &= p, \\
   1+\exp(-t) &= \frac{1}{p}, \\
   \exp(-t) &= \frac{1}{p} - 1, \\
   t &= -\ln\left(\frac{1}{p} - 1\right), \\
   t &= \ln\left(\frac{p}{1-p}\right).
   \end{aligned}$$

   On peut alors générer un échantillon de cette variable aléatoire avec le code suivant :

```r
sample<-function(n){
    # Stockage de l'échantillon de taille n
    ech = c()
    for (i in 1:n){
        u = runif(1)
        ech = c(ech,log(u/(1-u)))
    }
return(ech)
}

sample(3)
```

```
## [1] -0.73362548  2.27208817 -0.05654428
```

Exercice 6 : Un cas concret

Des personnes arrivent dans un magasin de chaussures. Chaque personne regarde un certain nombre de chaussures avant de décider quelle paire acheter.

1. Soit N le nombre de personnes qui entre dans le magasin en une heure. Sachant que $\mathbb{E}[N] = 10$, quelle serait une bonne distribution pour la variable aléatoire N ?

   La variable aléatoire consiste à compter le nombre de personnes qui entrent dans un magasin, on va donc effectuer un *processus de comptage*, processus qui utilise une loi de Poisson. On va donc modéliser N par une loi de Poisson de paramètre $\lambda = 10$. En effet, une loi de Poisson de paramètre λ vérifie $\mathbb{E}[N] = Var[N] = \lambda$.

2. Le client i essaie X_i paires de chaussures qu'il n'aime pas avant de trouver une paire qui lui convienne, *i.e.* $X_i \in \{0, 1, 2, \cdots\}$. Supposons que la probabilité p qu'ils aiment une paire de chaussures est de 0.8, indépendamment des autres paires regardées. Quelle serait une bonne distribution de X_i.

   On a déjà rencontré cette loi au premier TD, *i.e.* au TD 0, on pourrait modéliser cela par une loi Géométrique de paramètre p où p est la probabilité de succès, *i.e.* trouver une paire qui convienne. Ainsi :

   $$\mathbb{P}[X_i = k] = q^k p, \ \forall k \in \mathbb{N}.$$

3. Soit Y le nombre total de paires de chaussures qui ont été essayées, sans tenir compte de celles achetées. Supposons que chaque client agisse indépendamment des autres, donner une expression de Y en fonction de N et de X_i et écrire une

fonction permettant de simuler N, X_i et Y.

On a bien évidemment :

$$Y = \sum_{i=1}^{N} X_i$$

On peut montrer que cette variable aléatoire suit une *loi binomiale négative*. Y représente le nombre d'échecs avant l'obtention de N succès (chaque client repart avec une paire de chaussure).
On a alors

$$\mathbb{P}[Y = k] = \binom{k+N-1}{k} p^N q^k$$

La loi binomiale négative s'interprète comme la loi de probabilité de la variable aléatoire X qui compte le nombre d'échecs observés avant l'obtention de N succès pour une série d'expériences indépendantes, sachant que la probabilité d'un succès est p.

En terme de simulation, il faut simplement utiliser l'expression de Y, on peut donc le faire avec le code suivant :

```
sample<-function(n,lambda=10,p=0.8){
    # Stockage de l'échantillon de taille n
    ech = c()
    for (i in 1:n){
        N = rpois(1,lambda)
        # on génère autant de valeurs de X que l'on a de clients
        X = rgeom(N,p)
        Y = sum(X)
        ech = c(ech,Y)
    }
return(ech)
}
sample(3)
```

```
## [1] 3 1 2
```

On a fait le choix de ne pas repasser par la loi uniforme pour simuler les variables aléatoires qui suivent une loi de poisson et une loi géométrique.
Pour la loi de Poisson, nous aurions pu passer par l'exercice 3, en revanche, nous n'avons pas encore vu comment faire cela avec la loi géométrique, bien

que l'on connaisse sa fonction de répartition (–> Exercice !)

4. Quelle est la valeur de $\mathbb{P}[Y=0]$? Essayer de retrouver cette valeur à l'aide du code précédent.

$$
\begin{aligned}
\mathbb{P}[Y=0] &= \mathbb{P}\left[\sum_{i=1}^{N} X_i = 0\right], \\
&\quad \downarrow \text{ si la somme est nulle, cela signifie que chaque } X_i \text{ est égale à } 0 \\
&= \mathbb{P}\left[X_1 = 0, X_2 = 0, \cdots, X_N = 0\right], \\
&\quad \downarrow \text{ indépendance des variables aléatoires} \\
&= \mathbb{P}\left[X_1 = 0\right] \times \mathbb{P}\left[X_2 = 0\right] \times \cdots \times \mathbb{P}\left[X_N = 0\right], \\
&\quad \downarrow \ \mathbb{P}\left[X_i = 0\right] = p = 0.8 \\
&= p^N, \\
&= 0.8^N.
\end{aligned}
$$

Pour cette simulation, on va reprendre le code précédent, mais on va fixer la valeur de N.

```
sample<-function(n,p=0.8){
    # Stockage de l'échantillon de taille n
    ech = c()
    for (i in 1:n){
        N = 10
        # on génère autant de valeurs de X que l'on a de clients
        X = rgeom(N,p)
        Y = sum(X)
        ech = c(ech,Y)
    }
return(ech)
}
# On compare la valeur théorique à celle obtenue
# par échantillonnage
n = 1000
sum(sample(n)==0)/n
```

```
## [1] 0.11
```

```
0.8^10
```

```
## [1] 0.1073742
```