Lois de probabilités discrètes sur R

Julien Parfait Bidias Assala

2024-08-21

Background

- Les lois de probabilités discrètes en statistiques permettent de modéliser des situations où les variables aléatoires prennent un nombre fini ou dénombrable de valeurs distinctes. Ces lois sont couramment utilisées pour décrire des phénomènes où les résultats possibles sont limités à des catégories distinctes ou des événements discrets.
- Par exemple, la loi binomiale permet de modéliser le nombre de succès dans une série d'essais indépendants, tandis que la loi de Poisson est utilisée pour décrire le nombre d'événements survenant dans un intervalle de temps ou d'espace fixe. D'autres exemples incluent la loi géométrique, qui mesure le nombre d'essais avant le premier succès, et la loi hypergéométrique, utilisée pour des échantillons sans remise.

Rappel : créer une fonction dans R

Pour créer une fonction dans R, la syntaxe est donnée par le bout de code ci-dessous. Elle est relativement proche de celle utilisée sur python.

```
nom_de_la_fonction = function(arg1, arg2, ...) {
# Corps de la fonction
  resultat = arg1 + arg2  # Exemple de calcul
# On retourne le résultat
return(resultat)
}
```

On compile ensuite on appelle la fonction avec son nom et en passant à l'intérieur les différents arguments.

```
nom_de_la_fonction(2, 2)
```

```
[1] 4
```

Lois de probabilités discrètes

Comment modéliser simplement, sans packages et de façon tout à fait compréhensible toutes ces différentes lois ? Tel est l'objet de la présentation.

Nous présentons la loi :

- De Bernoulli
- Binomiale
- De Poisson
- Géométrique
- Uniforme discrète
- Hypergéométrique
- Binomiale négative

Loi de Bernoulli

X suit une loi de Bernoulli lorsque :

$$P(X = k) = p^k(1-p)^{1-k} \quad avec \quad k = 0, 1 \quad et \quad p \in]0, 1[$$

D'où le programme ci-dessous.

```
bernoulli = function(p,k){
  if ( (k==0 | k==1) & (p>0 & p<1) ){
        probabilite = p**k * (1-p)**(1-k)
        print(probabilite)
  } else
    print("Entrer des valeurs correctes de k ou de p")
}
```

Loi de Bernoulli

Supposons que X suit une loi de Bernoulli avec $p = 0.4$. On veut calculer $P(X = 0)$ et $P(X = 1)$.

```
bernoulli(0.4, 0)
```

```
[1] 0.6
```

```
bernoulli(0.4, 1)
```

```
[1] 0.4
```

```
bernoulli(0.4, 3)
```

```
[1] "Entrer des valeurs correctes de k ou de p"
```

Espérance, variance et Fonction de répartition

Pour la moyenne et la variance il suffit de prendre :

$$E(X) = p$$

$$Var(X) = pq$$

La construction de la fonction de répartition passe par la relation :

$$F_X(k) = P(X \leq k)$$

On peut passer aussi par la relation :

$$F_X(k) = P(X \leq k) = \sum_{i=0}^{k} p^k (1-p)^{1-k}$$

Ainsi, si $k = 0$ on a $1 - p$. Par contre, si $k = 1$ on aura 1.

Espérance, variance et Fonction de répartition

```
repartition.bern = function(p,k){
  i      = 0
  somme = 0
  while(i<=k){
    if ( (k==0 | k==1) & (p>0 & p<1) ){
      probabilite = p**i * (1-p)**(1-i)
      somme = somme + probabilite
      i = i+1
      } else
        print("Entrer des valeurs correctes de k ou de p")
  }
  print(somme)
}
```

Fonction de répartition

Application :

Calculons $P(X \leq 0)$ et $P(X \leq 1)$ pour une loi de Bernoulli de paramètre $p = 0.4$.

```
repartition.bern(0.4, 0)
```

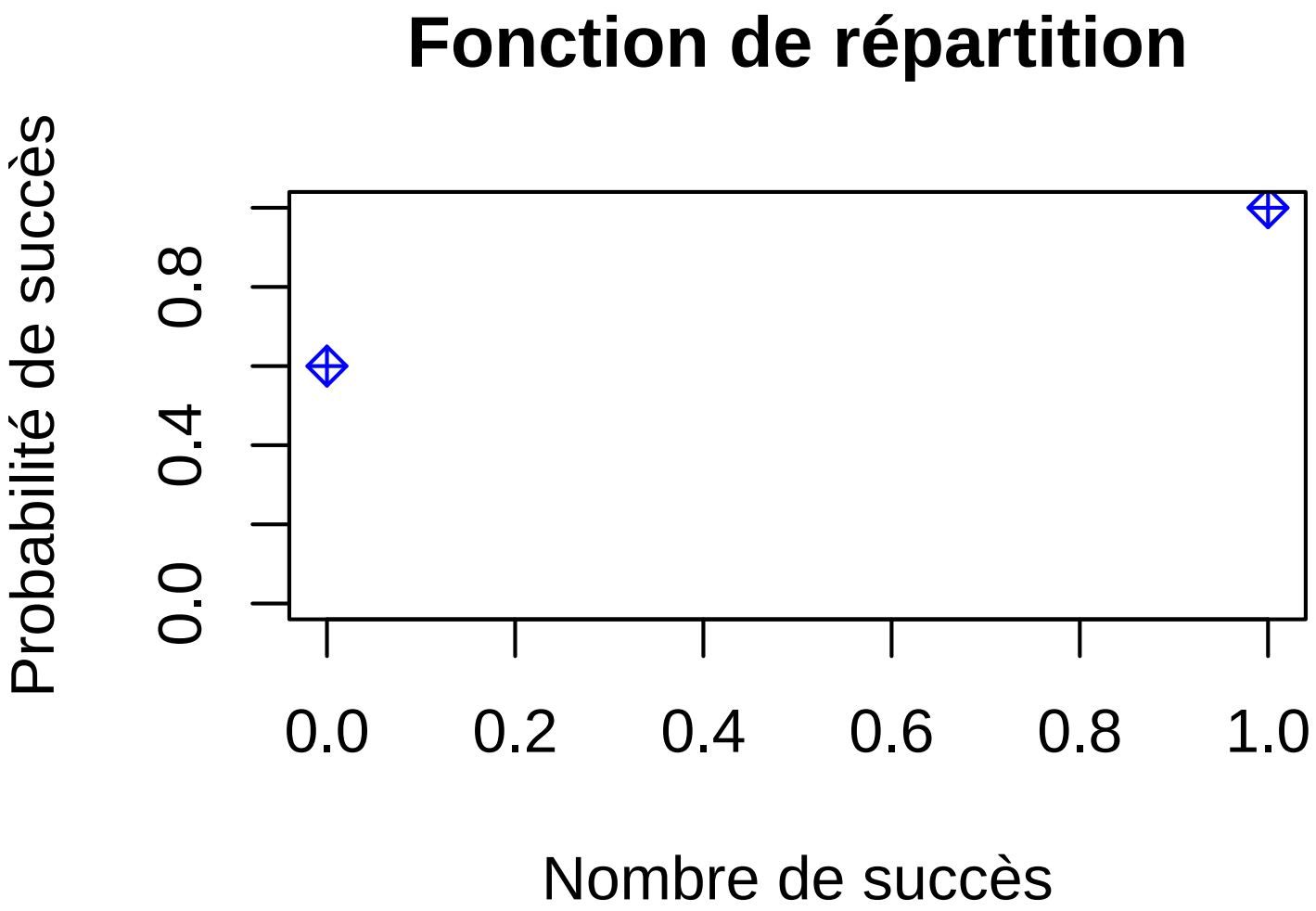
```
[1] 0.6
```

```
repartition.bern(0.4, 1)
```

```
[1] 1
```

La fonction de répartition associée à X prend ainsi deux valeurs quelque soit la valeur du paramètre p.

Fonction de répartition



Loi Binomiale

$P(X = k)$ pour une loi binomiale :
avec $k = 0, 1, \cdots, n$; $n \in N^*$ $p \in]0, 1[$; $k \leq n$

$$P(X = k) = C_n^k p^k (1 - p)^{n-k}$$

```
binome = function(n, p, k){
  if(k<=n & (p>0 && p<1) & n>0 && is.numeric(n) && n%%1==0){
    combinaison = factorial(n)/(factorial(k)*factorial(n-k))
    probabilite  = combinaison * p**k * (1-p)**(n-k)
    print(probabilite)
  } else{
    print("Entrer des valeurs correctes de n, p et k")
  }
}
```

Loi Binomiale

Application : $P(X = 1)$ pour une loi $B(n = 2, p = 0.5)$:

```
k = 1
n = 2
p = 0.5
binome(n, p, k)
```

```
[1] 0.5
```

Si on veut calculer par exmple $P(X = 4)$ avec X qui suit une loi $B(n = 2, p = 0.5)$ alors on aura :

```
binome(2, 0.5, 4)
```

```
[1] "Entrer des valeurs correctes de n, p et k"
```

Ce qui est tout a fait normal car k ne peut dépasser n.

Espérance, variance et fonction de répartition

$E(X) = np$ et $Var(X) = np(1-p)$ on peut s'amuser à programmer une petite fonction qui renvoit ces caractéristiques :

```
carac.binom = function(n, p){
  if(n>0 && (p>0 & p<1) && is.numeric(n) && n%%1==0){
      esperance = n*p
      variance  = n*p*(1-p)
      return(list(Esperance = esperance, Variance=variance))
  } else{
    print("Entrer des valeurs correctes de n ou de p")
  }
}
```

Exemple

si X suit une B(4, 0.8) alors :

```
carac.binom(4, 0.8)
```

```
Esperance
[1] 3.2

Variance
[1] 0.64
```

Pour la fonction de répartition, on peut passer la relation :

$$F_X(k) = P(X \leq k) = \sum_{i=0}^{k} P(X = i) = \sum_{i=0}^{k} C_n^i p^i (1-p)^{1-i}$$

Fonction de répartition de la loi Binomiale

```
repartition.binom = function(n, p, k){
  i = 0
  somme = 0
  if(k<=n & (p>0 & p<1) & n>0 & is.numeric(n) & n%%1==0){
    while(i<=k){
      combinaison=factorial(n)/(factorial(i)*factorial(n-i))
      probabilite= combinaison * p**i * (1-p)**(n-i)
      somme = somme + probabilite
      i = i + 1
      }
    }else{
    print("Entrer des valeurs correctes de n, de p ou de k")
    }
 print(somme)
}
```

Exemple

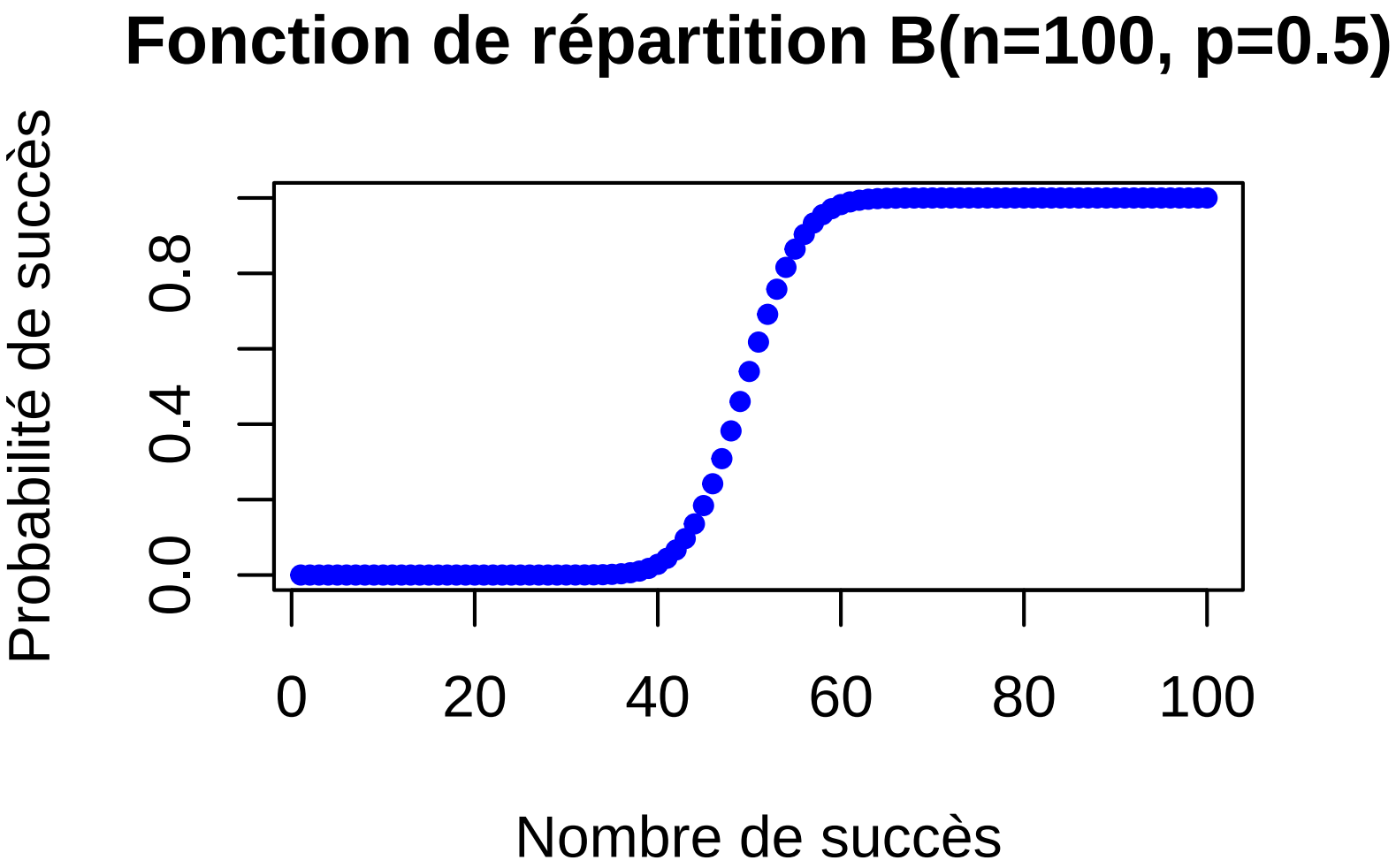
Application :

Calculons $P(X \leq 1)$ pour une loi $B(2, 0.5)$. On sait que $P(X \leq 1) = P(X = 0) + P(X = 1)$. Par conséquent, $P(X \leq 1) = 0.25 + 0.5 = 0.75$.

```
n = 2
p = 0.5
k = 1
repartition.binom (n, p, k)
```

```
[1] 0.75
```

Fonction de répartition d'une loi B(100, 0.5) :



Lois de Poisson

La P(X=k) lorsque X suit une loi de Poisson est donnée par :

$$P(X = k) = \frac{e^{-\lambda}\lambda^{k}}{k!}$$

Avec $k = 0, 1, \cdots, \infty$ et $\lambda > 0$. $E(X) = \lambda$ et $Var(X) = \lambda$.

```
poisson = function(lambda, k){
  if(lambda>0 && k>0 && is.numeric(k) && k%%1==0){
    probabilite = (exp(-lambda)*lambda**k)/factorial(k)
    print(probabilite)
  } else {
    print("Entrer des valeurs correctes de k ou de lambda")
  }
}
```

Exemple d'application

```
poisson(10, 2)
```

```
[1] 0.002269996
```

Pour n assez grand et p très petit, on peut approximer une loi binomiale par une loi de Poisson. Supposons que l'on veuille calculer : P(X=2) avec X suivant une loi $B(100, 0.01)$. On pose donc $np = \lambda$ et on calcule aussi P(X=2) avec X suivant une loi $P(\lambda = np)$

```
b1 = binome(100, 0.01, 2)
```

```
[1] 0.1848648
```

```
p1 = poisson(100*0.01, 2)
```

```
[1] 0.1839397
```

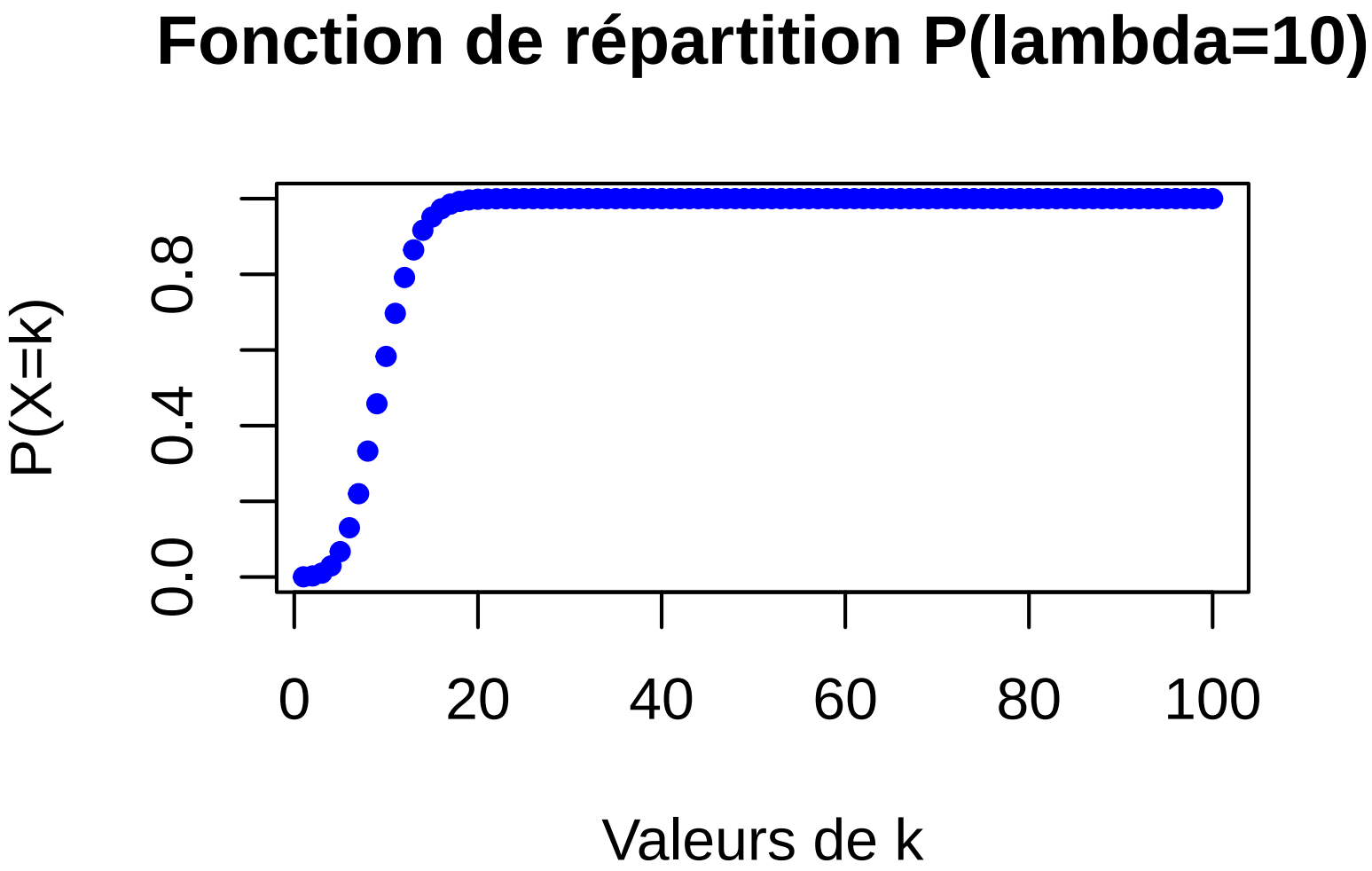
Fonction de répartition de la loi de Poisson

On applique la formule suivante :

$F_X(k) = P(X \leq k) = \sum_{i=0}^{k} P(X = i) = \sum_{i=0}^{k} \frac{e^{-\lambda}\lambda^i}{i!}$

```
repartition.poiss = function(lambda, k){
  i = 0
  somme = 0
  if(lambda>0 && k>0 && is.numeric(k) && k%%1==0){
    while(i<=k){
      probabilite = (exp(-lambda)*lambda**i)/factorial(i)
      somme = somme + probabilite
      i = i + 1
      }
    }else{
    print("Entrer des valeurs correctes de n, de p ou de k")
    }
 print(somme)
}
```

Fonction de répartition de la loi de poisson



Loi Géométrique

X suit loi géométrique si et seulement si sa loi de probabilité est donnée par :

$$P(X = k) = p(1 - p)^{k-1}$$

Avec $k = 1, 2, \cdots, \infty$ et $p \in]0, 1[$

```
geometrique = function(p, k){
  if(k>=0 &  (p>0 & p<1) &  is.numeric(k) & k%%1==0){
    probabilite = p*(1-p)**(k-1)
    print(probabilite)
  }else{
    print("Entrer des valeurs correctes de p ou de k")
  }
}
```

Exemple :

Calculons $P(X = 3)$ avec X suivant une loi $G(p = 0.7)$.

```
geometrique(0.7, 3)
```

```
[1] 0.063
```

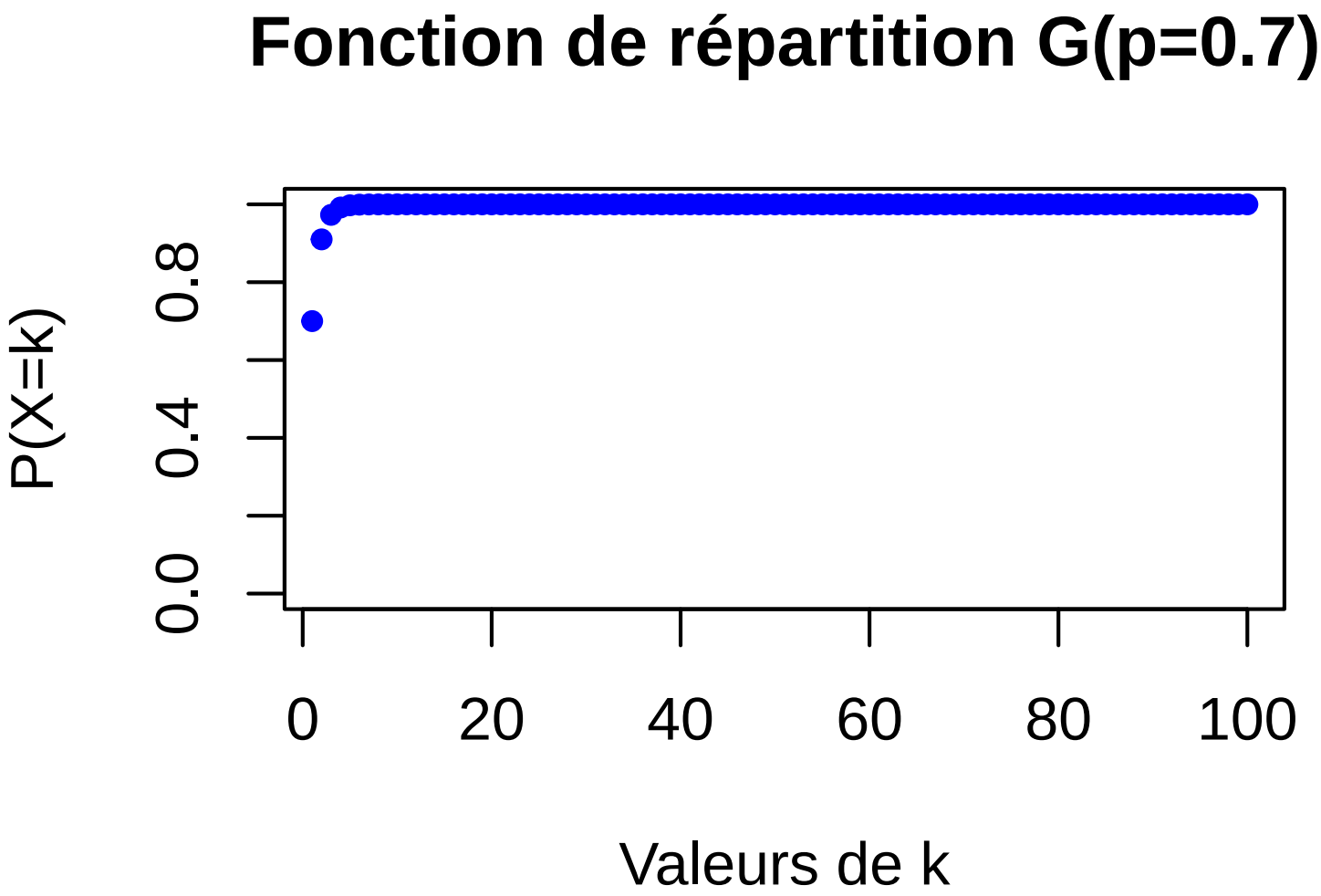
Les caractéristiques d'une loi géométrique sont : $E(X) = \frac{1}{p}$ et $Var(X) = \frac{1-p}{p^2}$.

Fonction de répartition de la loi géométrique

$$F_X(k) = P(X \leq k) = \sum_{i=0}^{k} P(X = i) = \sum_{i=0}^{k} p(1-p)^{i-1}$$

```
repartition.geom = function(p, k){
  i = 1
  somme = 0
  if(k>=0 &  (p>0 & p<1) &  is.numeric(k) & k%%1==0){
    while(i<=k){
      probabilite = p*(1-p)**(i-1)
      somme = somme + probabilite
      i = i + 1
      }
    }else{
    print("Entrer des valeurs correctes de p ou de k")
    }
 print(somme)
}
```

Fonction de répartition de la loi géométrique



Loi uniforme discrète

Soit X une variable aléatoire discrète uniforme prenant n valeurs $k_1, k_2, \ldots, k_n$. La fonction de probabilité de X est donnée par :

$$P(X = k_i) = \frac{1}{n} \quad \text{pour } k_i \in \{k_1, k_2, \ldots, k_n\}.$$

```
uniforme = function(n){
  if(n>0 & is.numeric(n) & n%%1==0){
    probabilite = 1/n
    print(probabilite)
  } else{
    print("Entrer une valeur correcte de n")
  }
}
```

Exemple

Considérons le jet d'un dé non biaisé. L'ensemble des $n = 6$ valeurs possible de X est $A = 1, 2, 3, 4, 5, 6$. A chaque fois que le dé est jeté, la probabilité d'un résultat donné vaut $1/6$.

```
n=6
uniforme(n)
```

```
[1] 0.1666667
```

Les caractéristiques de la loi uniforme discrète : $E(X) = \frac{n+1}{2}$. La variance : $Var(X) = \frac{n^2-1}{12}$

Fonction de Répartition de la loi uniforme discrète

La fonction de répartition $F(k)$ pour X est définie comme suit :

$$F(k) = \begin{cases} 0 & \text{pour } k < 1, \\ \frac{\lfloor k \rfloor}{n} & \text{pour } 1 \leq k < n, \\ 1 & \text{pour } k \geq n. \end{cases}$$

Exemple :

Considérons le jet d'un dé non biaisé. L'ensemble des $n = 6$ valeurs possible de X est $A = 1, 2, 3, 4, 5, 6$. A chaque fois que le dé est jeté, la probabilité d'un résultat donné vaut $1/6$. calculons $P(X \leq 4)$. Cela revient à calculer $F(4) = 4/6 = 2/3 = 0.66667$.

Fonction de Répartition de la loi uniforme discrète

```
repartition.unif.d = function(n, k){
  if((k>=1 & k<n) & n>0 & is.numeric(n) & n%%1==0){
    probabilite = floor(k)/n
    print(probabilite)
  } else{
    if(k<1){
      print(0)
    } else{
      if(k>=n){
        print(1)
      }else{
        print("Entrer une valeur correcte de n ou de k")
      }
    }
  }
}
```

Application à l'exemple précédent

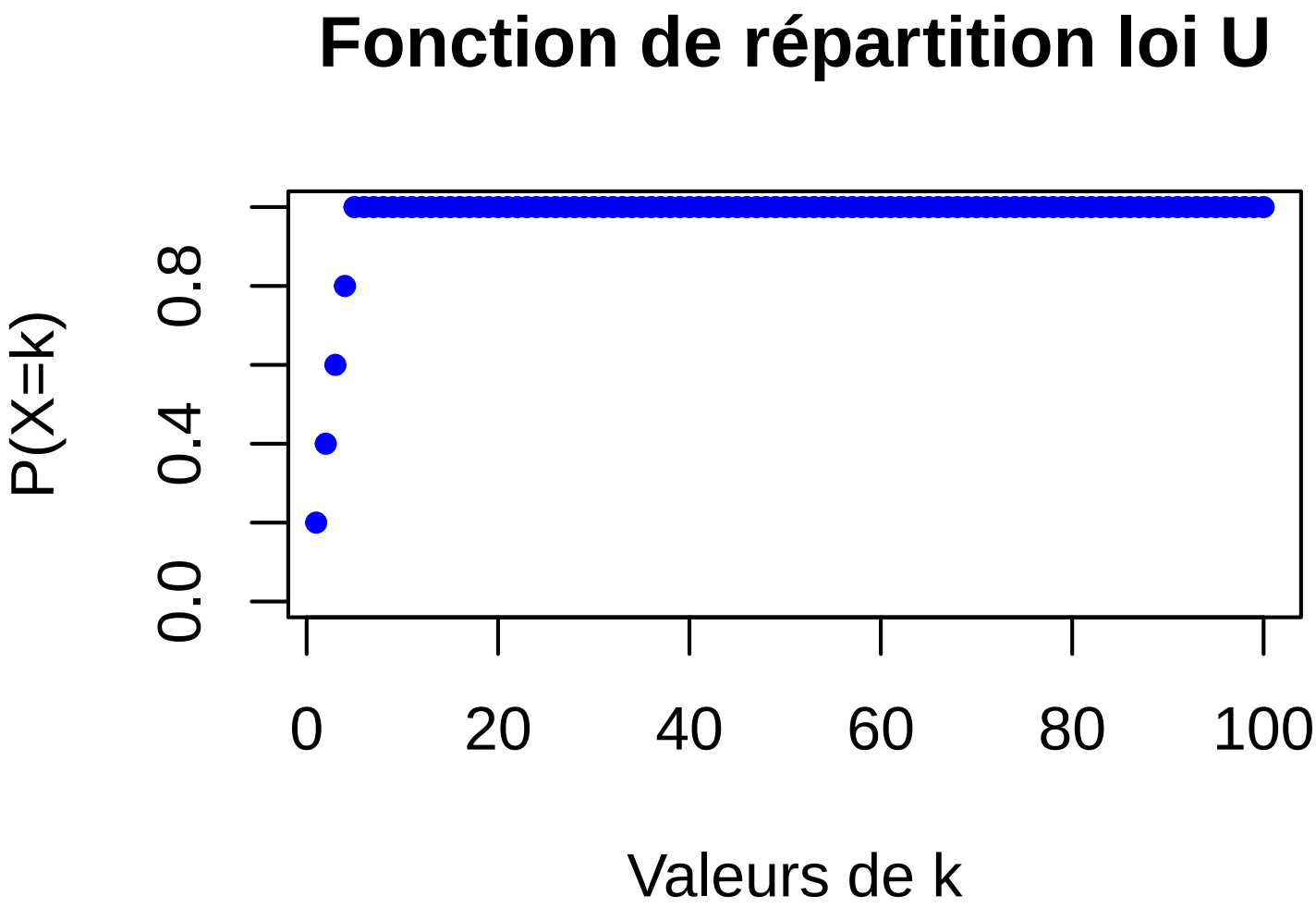
```
repartition.unif.d(6, 4)
```

```
[1] 0.6666667
```

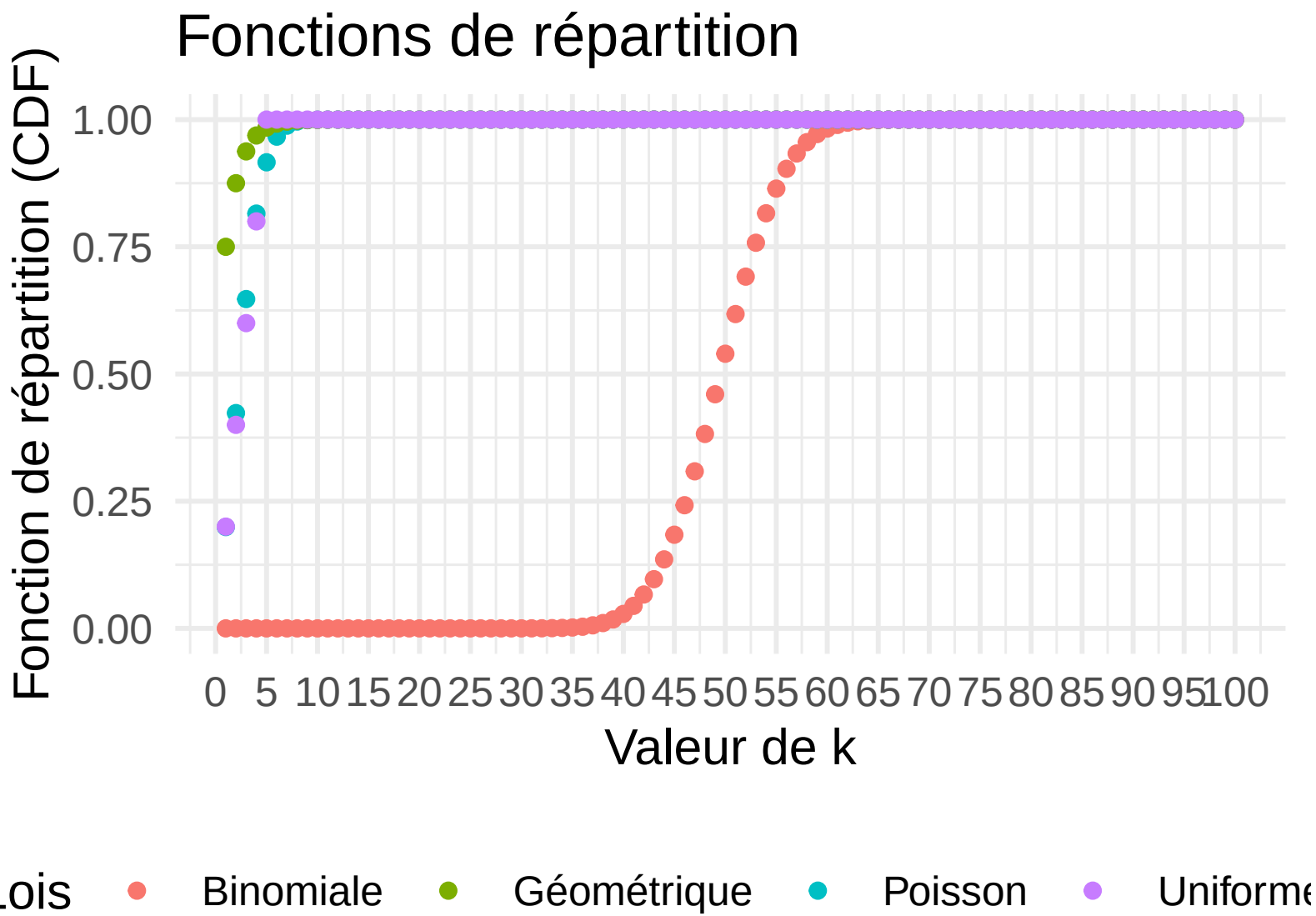
Il y a donc 2 chances sur 3 que les valeurs des faces du dé soient toutes inférieures ou égales à 4 (soit 67%).

Enfin, tu peux tout simplement t'inspérer de tout cet arsenal pour programmer les autres lois.

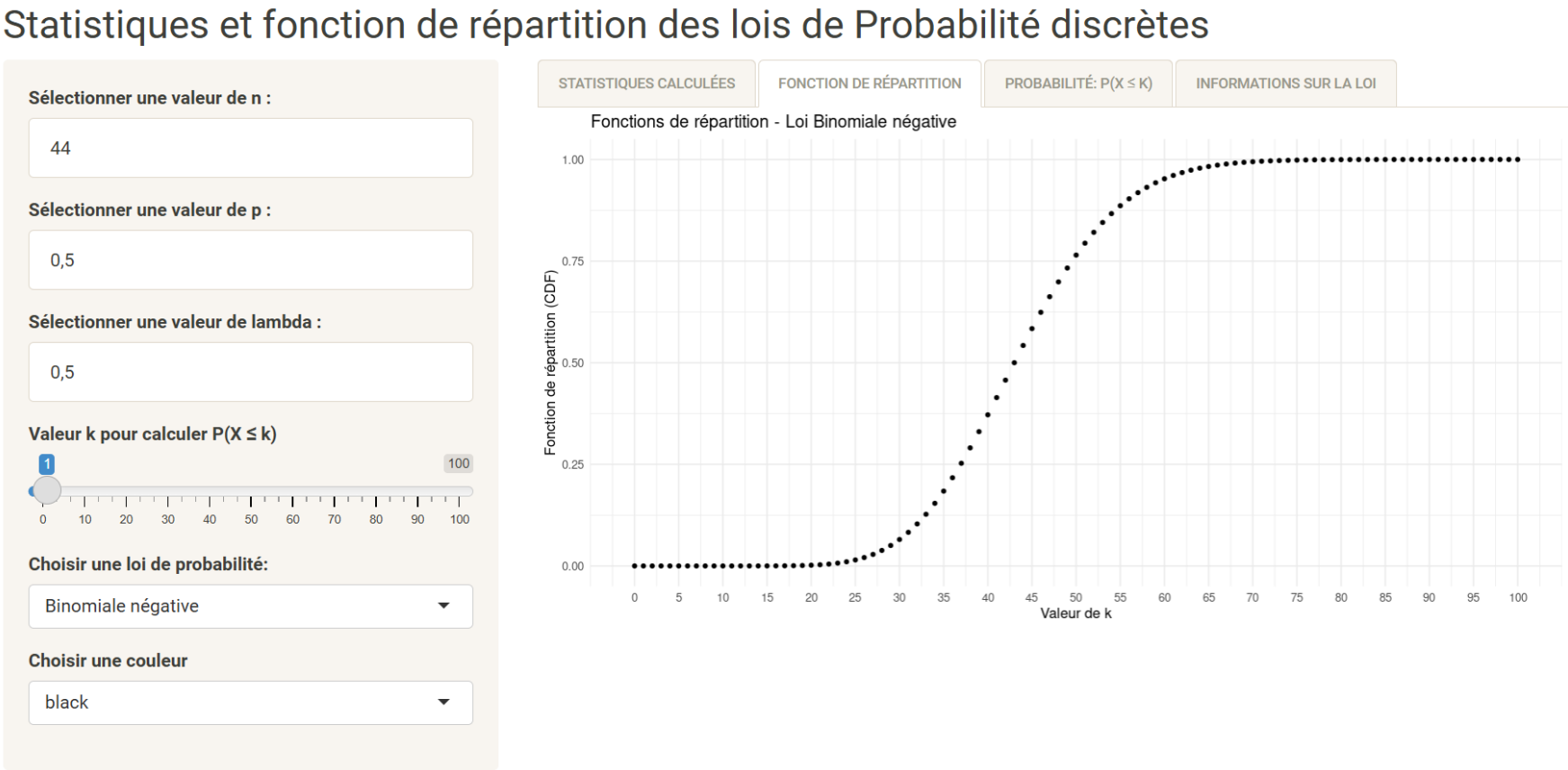
Fonction de répartition de la loi uniforme



Résumé


Fonctions de répartition
Fonction de répartition (CDF)
1.00
0.75
0.50
0.25
0.00
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
Valeur de k
Lois
Binomiale
Géométrique
Poisson
Uniforme

Application Shiny



Lien de l'application :
https://julienparfaitbds.shinyapps.io/Lois-de-probabilites-discretes/

Fin

julienbidias246@gmail.com

Bonne lecture et surtout n'hésite pas à écrire au mail ci-dessus pour faire des suggestions et avoir aussi le code complet avec tous les commentaires à l'appui.

A suivre...