# Rapport_Module_Probabilité

SASIKUMAR BA ZIHOUNE GUENFICI MENDES

01/12/2022

# Module de Probabilité

Dans le cadre de la SAE 3.01, nous devions implementer une application web de calcul de probabilités dans le cadre d'une loi normale de paramètres m et σ. Pour ce faire, nous avons choisis de coder en python pour la partie modèle et JavaScript,PHP et HTML pour l'interface graphique.

## 1.1 Introduction

Nous avons utilisé les propriétés suivantes:

- La loi normale suivant une loi de Gauss, sa courbe est en cloche et son pic est situé à l'espérance (noté σ). Notre objectif est d'obtenir la probabilité P(X < t), autrement dit P(-inf < X < t).
- Dans le cas où t>σ (cas 1), on a P(X < t) = P(-inf < X < σ) + P(σ < X < t) = 0.5 + P(σ < X< t) (voir Figure 1). Il nous reste donc plus que P(σ < X < t) à calculer, ce qui nous est possible grâce aux formules fournies
- Dans le cas où σ>t (cas 2), notre méthode à été de calculer "l'inverse" de P(X < t) c'est à dire P(X > t).
  Ainsi, P(X > t) = P(t < X < σ) + P( σ < X < +inf )= P(t < X < σ) + 0.5 (voir Figure 2).
  Ou plus simplement : P(X > t) = 1 - P(X < t) (voire Figure 1 et 2)
  Il ne nous reste donc plus que P(t < X < σ) à calculer.

On en déduit également que si σ=t (cas 3), P(X < t) = P(X < σ) = 0.5 (voir Figure 3).

De plus, nos fonctions étant des approximations, quelques précautions ont dû être mises en place pour éviter des résulats incohérents (P<0 ou P>0).

```
if res < 0 :
  res = 0
if res > 1:
    res = 1
    return res
```

## 1.2 Codage des méthodes

A noter que nous utilisons les bibliothèques math (notamment pour π) et numpy.

```
from math import *
from numpy import arange
```

**Loi Normale :**

```
def loi_normale(x, m, et):
    """
    Fonction de la loi normale telle que donnée en cours
    Entrées :
            x : variable (float / int)
            m : éspérance (float / int)
            et : écart-type (float / int)
    Retour : res : f(x) (float / int)
    """
    denom = et * sqrt(2 * pi)
    e = exp((-1 / 2) * ((x - m) / et) ** 2)
    res = e / denom

    return res
```

Pour toutes les méthodes (sauf celle de Simpson), nous travaillons sur l'intervalle [σ,t] ou [σ,t].

#### 1.2.1 La méthode des rectangles droits

$$\int_a^b f(t)dt \approx \frac{b-a}{n} \sum_{k=1}^{k=n} f\left(a_k\right)$$

Formule Fournie

Le calcul de la somme :

```
a = m # l'éspérance
b = t # l'écart-type
pas = (b-a)/n #distance entre 2 division
if a == b :
    return 0.5
for a in arange(a,b,pas): # la somme
    sum += loi_normale(a, m, et)
res = sum*pas + 0.5
return res
```

### 1.2.2 La méthode des rectangles gauches

$$\int_a^b f(t)dt \approx \frac{b-a}{n} \sum_{k=0}^{k=n-1} f\left(a_k\right)$$

Formule Fournie

Le calcul de la somme :

```
  a = m
  b = t
  pas = (b-a)/n
  if a == b :
      return 0.5
  for k in arange(a+pas,b+pas,pas):
      sum += loi_normale(k, m, et)
  res = sum*pas + 0.5
  return res
```

La boucle va de a+pas à b inclus car la somme va de n=1 à n inclus.

### 1.2.3 La méthode des rectangles médians

$$\int_a^b f(t)dt \approx \frac{b-a}{n} \sum_{k=0}^{k=n-1} f\left(\frac{a_k + a_{k+1}}{2}\right)$$

Formule Fournie

Le calcul de la somme :

```
  a = m
  b = t
  pas = (b-a)/n
  if a == b :
      return 0.5
  for k in arange(a, b, pas):
      c = (k+k+pas )/2 #k = ak et k+pas = ak+1
      h = loi_normale(c, m, et)  # * i
      sum+= h
  res = sum*pas + 0.5 #
  return res
```

### 1.2.4 La méthode des trapèzes

$$\int_a^b f(t)dt \approx \frac{b-a}{2n}\left(f(a) + f(b) + 2\sum_{k=1}^{k=n-1} f\left(a_k\right)\right)$$

Formule Fournie

Le calcul de la somme :

```
a = m
b = t
pas = (b-a)/n
fa = loi_normale(a,m,et)
fb = loi_normale(b,m,et)
if a == b :
    return 0.5
for k in arange(a+pas, b, pas): de
    sum+= loi_normale(k,m,et)

res = ((b-a) / (2*n) * (fa + fb + 2*sum)) + 0.5
return res
```

### 1.2.5 La méthode de Simpson

$$\int_a^b f(t)dt \approx \frac{b-a}{6n}\left(f(a) + f(b) + 2\sum_{k=1}^{k=n-1} f\left(a + \frac{(k)(b-a)}{n}\right) + 4\sum_{k=0}^{k=n-1} f\left(a + \frac{(2k+1)(b-a)}{2n}\right)\right)$$

Formule Fournie

Pour cette dernière méthodes, travailler sur l'intervalle [σ,t] ne nous à pas réussit. Nous avons donc appliqué à la lettre la formule donnée dans le sujet.

Le calcul de la somme :

```
a = m
b = t
fa = loi_normale(a,m,et)
fb = loi_normale(b,m,et)

if a == b :
    return 0.5
for k1 in arange(1,n): # de 1 à n-1
    e1 = a + (k1*(b-a)) / n
    sum1+= loi_normale(e1,m,et)

for k2 in arange(n): # de 0 à n-1
    e2 = a + ((2*k2 +1) * (b-a)) / (2 * n)
    sum2+= loi_normale(e2,m,et)

res = ((b-a)/(6*n)) * (fa + fb + 2*sum1 + 4*sum2) + 0.5
return res
```

## 2 Échantillons d'exemples :

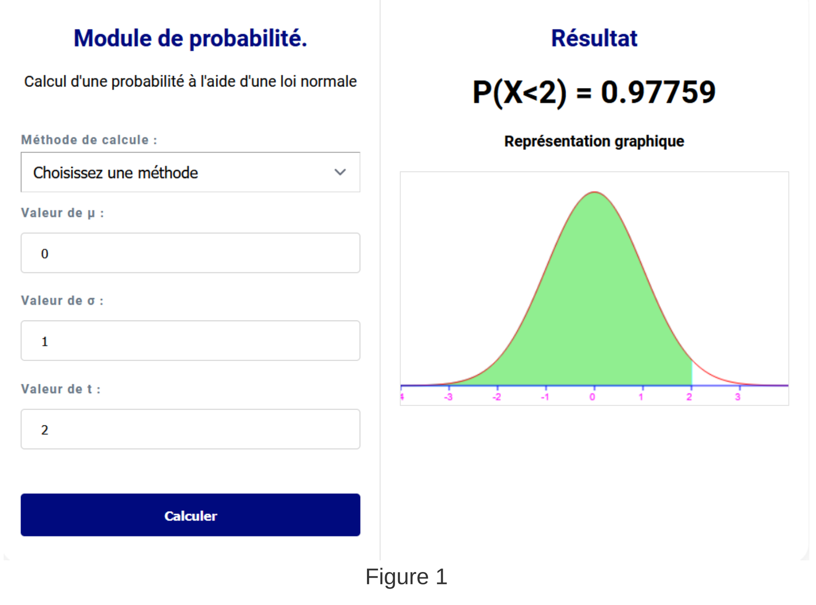


Figure 1

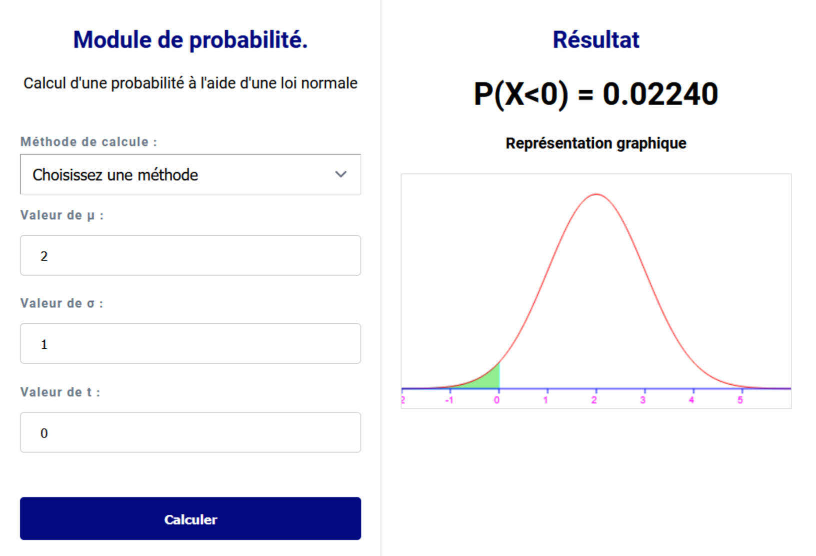


Figure 2

On peut remarquer que 0.97759 + 0.02240 = 0.99999 ~= 1 (cas 1 et 2)

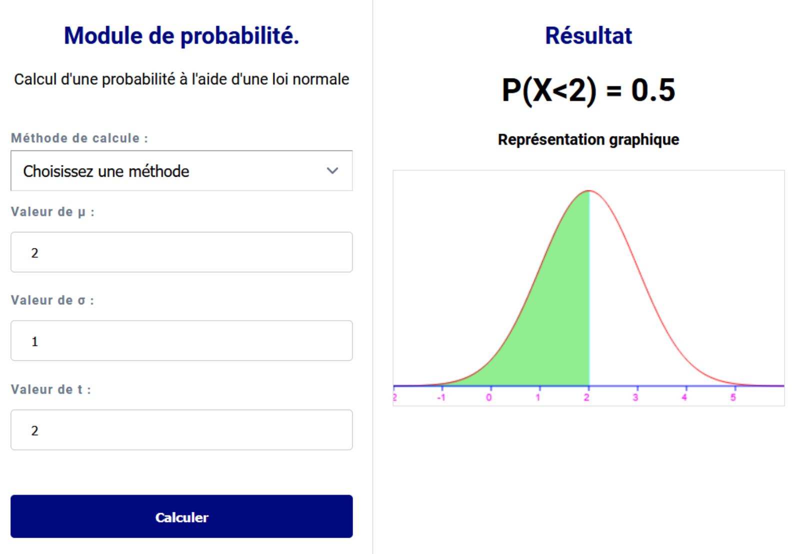


Figure 3 (cas 3 )