INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

Projet MSRO
GM3 2022 - 2023

Modélisation et Simulation de variable aléatoire uniforme

Étudiants :
KESSLER Aymeric
LANGOLFF Clément

Enseignant-responsable du projet :
Ciotir Ioana

Table des matières

Introduction

Dans le monde des probabilités, toutes les variables aléatoires ne suivent pas des lois connues, il est parfois difficile voir impossible de trouver leur loi. Cependant les lois usuelles peuvent nous permettre d'approcher ces lois. L'objectif de ce projet est de manipuler la loi uniforme pour simuler des variables aléatoires dont on ne connait que la fonction de densité. Nous essaierons ensuite de toucher à d'autres loi pour améliorer la qualité du modèle.

Première partie

Simulation d'une loi uniforme sur le disque unité

0.1 Loi uniforme dans le disque unité

Considérons (X,Y) un couple de variable aléatoire suivant une loi uniforme sur l'intervale $[-1,1]$. L'objectif est de simuler une loi uniforme sur un disque, c'est à dire tel que $X^2+Y^2 \leq 1$. Pour cela, nous considérons des couples de coordonnées (x,y) dont les valeurs sont prises par les variables aléatoire X et Y dans un carré unité. Si un couple de coordonnée vérifie $x^2+y^2 \leq 1$ alors nous acceptons le point. L'objectif est de vérifier que la loi du point obtenue suit bien une loi uniforme sur le disque.

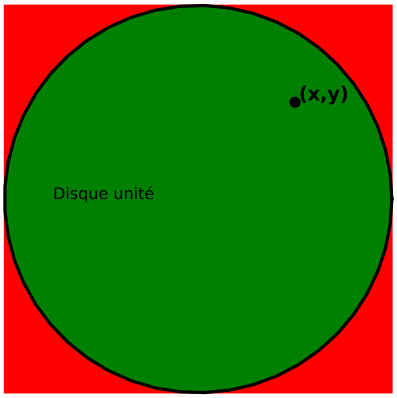
Si (X,Y) suit une loi uniforme sur le disque unité alors

$$f_{X,Y}(x,y) = \begin{cases} c & x^2+y^2 \leq 1 \\ 0 & sinon \end{cases}$$

De plus, nous devons avoir $P(\Omega) = \iint_\Omega f_{X,Y}(x,y)dxdy = 1$.

Or $\iint_\Omega f_{X,Y}(x,y)dxdy = \iint_{x^2+y^2\leq 1} cdxdy = c * Aire(\mathscr{B}(0,1)) = c\pi$. Ainsi $c = \frac{1}{\pi}$.

simulation de la loi uniforme sur le disque unité



A partir de la loi uniforme sur le disque unité, nous pouvons en déduire les lois marginales de X et Y. En effet, remarquons d'abord que la fonction de densité $f_{XY}(x,y)$ peut s'écrire sous la forme

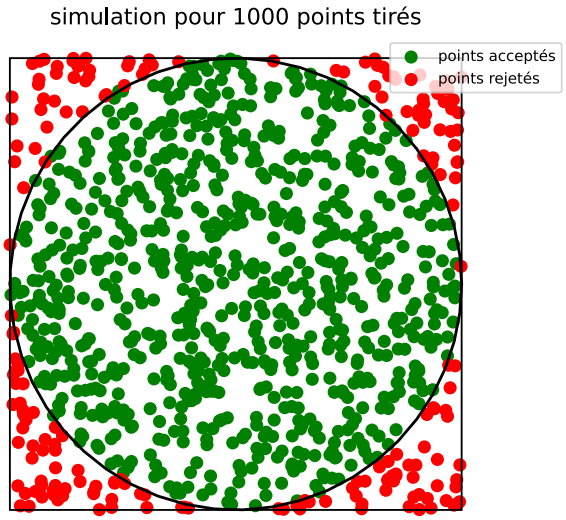
$$f_{XY}(x,y) = \frac{1}{\pi} 1_{[-1,1]}(x) 1_{[-\sqrt{1-x^2},\sqrt{1-x^2}]}(y)$$

Alors la loi marginal de X est donnée par

$$f_X(x) = \int_{-\infty}^{\infty} f_{XY}(x,y)dy = \frac{1}{\pi} 1_{[-1,1]}(x) \int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} dy = \frac{2\sqrt{1-x^2}}{\pi} 1_{[-1,1]}(x)$$

et de même la loi marginale de Y :

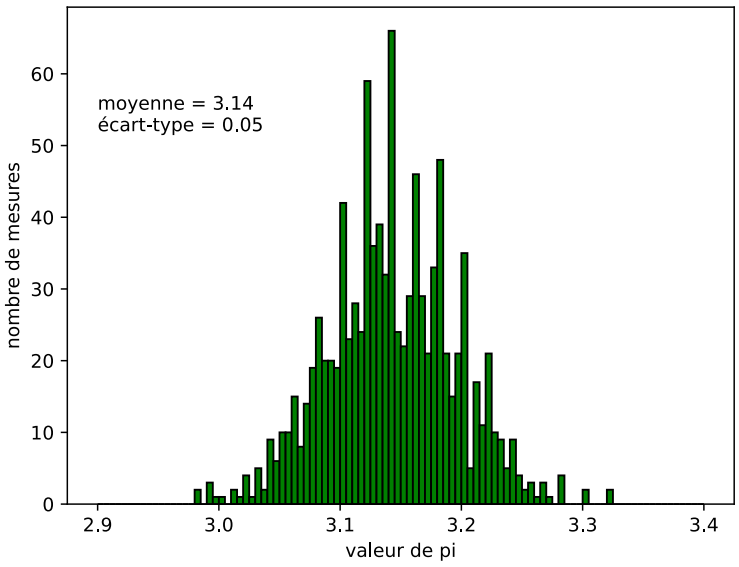
$$f_Y(y) = \frac{2\sqrt{1-y^2}}{\pi} 1_{[-1,1]}(y)$$



0.2 A la recherche de π

La probabilté d'avoir un point sur le disque unité est donnée par $P((x,y) \in \mathscr{B}(0,1)) = \frac{Aire(\mathscr{B}(0,1))}{Aire(carré)} = \frac{\pi}{4}$. Ainsi en comptant le nombre de points tiré dans le disque, en le multipliant par quatre fois et en divisant par le nombre de points tiré au total, nous devrions obtenir un valeur approximative de π.

FIGURE 1 – Valeur expérimental de π pour 1000 tirages



Deuxième partie

Algorithme de rejet

0.3 Transformation linéaire de la loi uniforme

Si X suit une loi uniforme sur [0,1] alors montrons que $Y = \alpha X + \beta$ suit une loi uniforme sur $[\beta, \alpha + \beta]$ avec $\alpha > 0$ et β un réel quelconque.
Rappelons que

$$F_X(x) = \begin{cases} x & \forall x \in [0,1] \\ 0 & sinon \end{cases}$$

$$\begin{aligned} F_Y(x) &= P(Y \leq x) \\ &= P(\alpha X + \beta \leq x) \\ &= \underbrace{P(X \leq \frac{x-\beta}{\alpha})}_{\alpha > 0} \\ &= F_X(\frac{x-\beta}{\alpha}) \end{aligned}$$

Remarquons alors que si $x \in [0,1]$ alors $y = \alpha x + \beta \in [\beta, \alpha + \beta]$. Et

$$F_Y(x) = \begin{cases} \frac{x-\beta}{\alpha} & \forall x \in [\beta, \alpha+\beta] \\ 0 & sinon \end{cases}$$

Donc Y suit bien une loi uniforme sur $[\beta, \alpha + \beta]$.

Pour la suite, nous souhaitons tirer un point de coordonnées (X,Y) uniformément distribué dans un rectangle de dimension [a,b]×[0,M].
Soit U_1 et U_2 deux lois uniforme sur [0,1].
L'objectif est de tirer des points uniformément entre a et b en se servant de U_1. Comme nous l'avons vu, une transformation affine de U_1 suffit pour avoir nos points.
En posant $\beta = a$ et $\alpha + \beta = b$, nous avons $\alpha = b - a$. Ainsi pour générer les points, il nous suffit de tirer une coordonnée x aléatoirement avec la loi uniforme U_1 et la formule $x' = a + (b-a)x$ nous donnera une coordonnée x' dans [a,b].
De même, pour avoir une coordonnée dans [0,M], nous posons $\beta = 0$ et $\alpha = M$. De la même manière, nous pouvons tirer une coordonnée y' dans l'intervale [0,M] par la formule $y' = My$ où y est une coordonnée tiré par la loi uniforme U_2.

L'idée de la méthode de rejet est d'utiliser un tirage d'une variable aléatoire connue (uniforme, normale...ect) pour pouvoir simuler une fonction à densité trop complexe, comme une fonction polynômiale.

Dans l'algorithme de rejet, la variable aléatoire X ainsi obtenu a pour densité f où f est la fonction de densité de la variable aléatoire que nous voulons simuler. Pour le démontrer, remarquons que X est une loi conditionnelle sur Y. En effet,

$$\begin{aligned} P(X \leq x_0) &= P(X \leq x_0 \mid Y \leq f(X)) \\ &= \frac{P(X \leq x_0 \cap Y \leq f(X))}{P(Y \leq f(X))} \end{aligned}$$

Or

- $P(X \leq x_0 \cap Y \leq f(X))$ correspond à la probabilité que notre point se trouve dans le rectangle de dimension $[a, x_0] \times [0, f(x_0)]$. Donc $P(X \leq x_0 \cap Y \leq f(X)) = \int_a^{x_0} \int_0^{f(x_0)} \frac{1}{M(b-a)} dydx$ car X et Y suivent une loi uniforme.
- $P(Y \leq f(X))$ est une probabilité à une dimension correspondant à la probabilité d'avoir l'image du point tiré par la fonction f dont l'univers est l'intervale $[0, M]$. Donc $P(Y \leq f(X)) = \int_0^{f(x_0)} \frac{1}{M} dy$.

Finalement

$$\begin{aligned} \frac{P(X \leq x_0 \cap Y \leq f(X))}{P(Y \leq f(X))} &= \frac{\int_a^{x_0} \int_0^{f(x_0)} \frac{1}{M(b-a)} dydx}{\int_0^{f(x_0)} \frac{1}{M} dy} \\ &= \frac{x_0 - a}{b - a} = F(x_0) \end{aligned}$$

0.4 Algorithme de rejet

— Tirer un point aléatoirement entre 0 et 1 grâce à la loi uniforme du langage
— Calculer la coordonnée $x_0 = a + (b-a)u_1$ qui suit la loi uniforme sur $[a, b]$ où u_1 est une variable tirée aléatoirement dans $[0, 1]$
— Tirer un point aléatoirement entre 0 et 1 grâce à la loi uniforme du langage
— Calculer la coordonnée $y_0 = Mu_2$ qui suit la loi uniforme sur $[0, M]$ où u_2 est une variable tirée aléatoirement dans $[0, 1]$
— Si $y_0 < f(x_0)$ on garde le point $A(x_0, y_0)$ sinon on le rejète

0.5 Exemple d'application pour une fonction polynomiale

Pour la suite, nous prendrons comme exemple la fonction $f(x) = 6x(1-x)1_{(0\leq x\leq 1)}$.
Vérifions tout d'abord que f est bien une densité de probabilité.

$$\begin{aligned}\int_{-\infty}^{+\infty} f(x)dx &= \int_0^1 6x(1-x)dx \\ &= 6[\frac{x^2}{2} - \frac{x^3}{3}]_0^1 \\ &= 6 * (\frac{1}{2} - \frac{1}{3}) = 1\end{aligned}$$

Donc f est bien une densité de probabilité.

Il nous faut maintenant définir notre rectangle de dimension $[a, b] \times [0, M]$.
Remarquons que plus la taille du rectangle est grand, plus la probabilité d'avoir un point (tiré dans le rectangle) sous la courbe est faible. En effet, augmenter l'aire du rectangle revient à agrandir l'univers Ω. L'objectif est donc de réduire au maximum cette aire et d'avoir une loi qui se rapproche au mieux de notre fonction.

Pour notre exemple, nous prenons la loi uniforme pour « envelopper » notre fonction. La fonction f étant définit sur l'intervale $[0, 1]$, nous pouvons tirer la coordonnée x du point juste avec une loi uniforme sur [0,1]. Le nombre M minimisant alors l'enveloppe de f est le maximum de la fonction f, c'est à dire M = $\frac{3}{2}$.

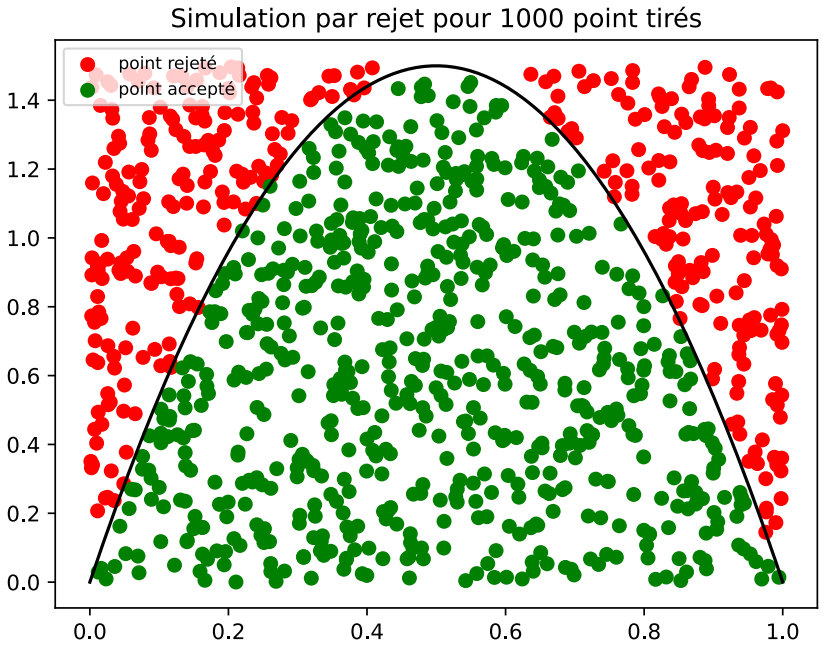
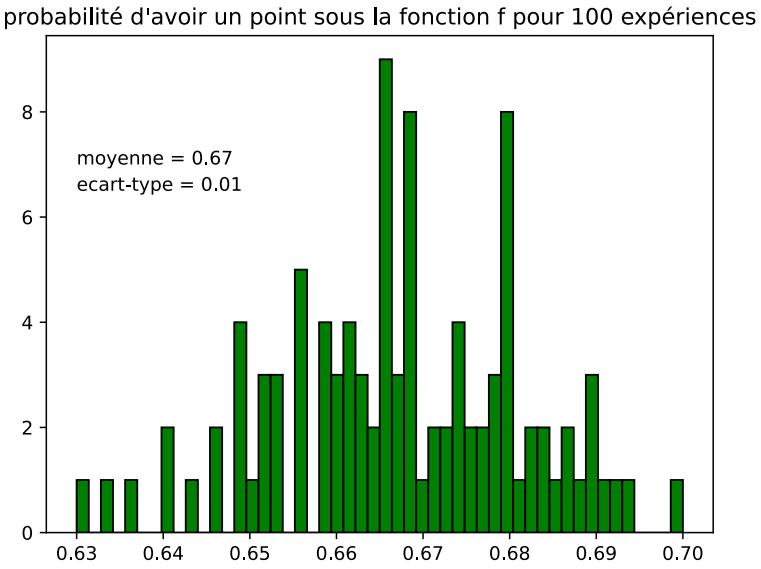


FIGURE 2 – Application de l'algorithme de rejet pour la fonction $f(x) = 6x(1-x)1_{(0\leq x\leq 1)}$

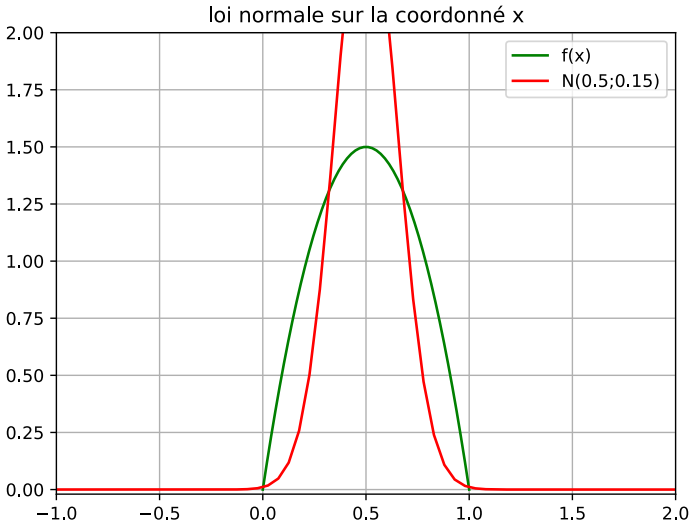
Pour mesurer la qualité du modèle, nous calculons la probabilité d'avoir un point sous la courbe f. C'est à dire que nous divisons le nombre de point accepté par le nombre de point totale. Nous réalisons plusieurs fois l'expérience afin d'avoir une probabilité moyenne.



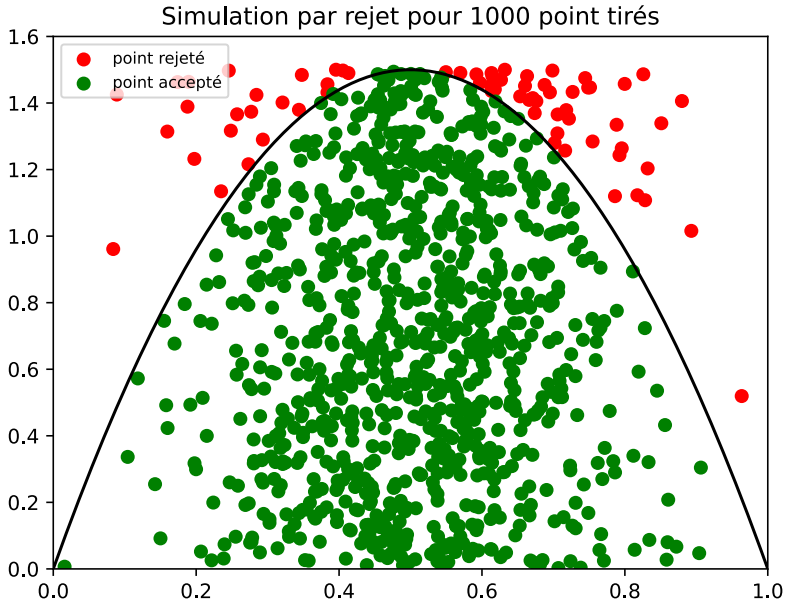
La probabilité d'avoir un point accepté est de 0,67 ce qui est une assez bonne approximation.

0.6 Vers une amélioration du modèle

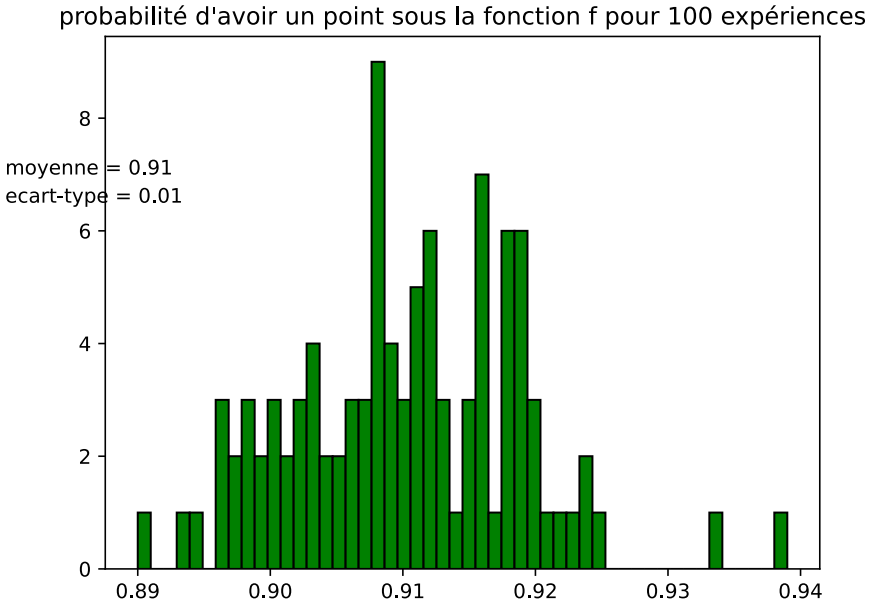
Pour améliorer notre modèle, nous pouvons mieux tirer nos points. En effet, en prenant une loi uniforme sur x, la probabilité que le point soit accepté est beaucoup plus grande lorsque la coordonnée x se trouve proche de l'antécédent du maximum de la fonction (noté a) (la probabilité est même de 1 lorsque la coordonnée x vaut a). Cette probabilité diminue lorsque l'on s'éloigne de a (la probabilité est même nulle en 0 et 1). Ainsi, en prenant une loi normale pour tirer la coordonnée x d'un point, nous augmentons nos chance d'avoir un point accepté.



En prenant $\mu = 0.5$ et $\sigma = 0.15$, nous avons une densité de probabilité qui correspond aux conditions citées précédemment.



Nous avons maintenant beaucoup plus de points sous la courbe f.



La moyenne d'avoir un point sous la courbe est de 0.91 ce qui est bien meilleur qu'en prenant une loi uniforme sur la coordonnée x du point.

Conclusion et perspectives

Ce projet nous a permis de pouvoir découvrir d'une autre façon les variables aléatoires. Nous avons pu découvrir des algorithmes qui sont très utiles dans le domaine des statistiques et les mettre en application avec python nous a donné une perception assez interessante de la théorie des probabilités.

Le perspectives pour ce projet sont assez larges, nous pouvons par exemple, étudier des variables aléatoires ayant des fonctions de densité beaucoup plus complexes qu'une fonction polynomiale. Nous pouvons encore optimiser le modèle en choisissant la bonne loi pour simuler la coordonnée y. Une autres étude intéressante serait de continuer la simulation sur des espaces beaucoup plus grands que l'espace à 2 dimensions.

Bibliographie

Livres
— La physique du sup en applications avec Python. Auteurs : Carpentier Renaud, Dépret Benoît. Edtion Ellipse.
— Monte Carlo Statical Methods. Auteurs : Christian P.Robert, George Casella. Edition Springer.

Annexes

Code Python pour simuler la loi uniforme sur le disque unité.

```
import numpy as np
import matplotlib.pyplot as plt

N = 1000
def traceCercle() :
        angle = np.linspace(0 , 2 * np.pi , 50)
        plt.plot(np.cos(angle) , np.sin(angle) , color = 'black')

def traceCarré(ax) :
        carré = plt.Rectangle((-1,-1),2,2, edgecolor = "black", fill = False)
        ax.add_patch(carré)

x = np.random.uniform(-1,1,N)
y = np.random.uniform(-1,1,N)
x_ext = []
x_int = []
y_ext = []
y_int = []

for i in range (N) :
    if (x[i]*x[i] + y[i]*y[i]) <= 1 :
        x_int.append(x[i])
        y_int.append(y[i])
    else :
        x_ext.append(x[i])
        y_ext.append(y[i])


proba = (len(x_int) / len(x))
print(proba)
fig , ax = plt.subplots()
plt.scatter(x_int, y_int, color = 'green',label = "points acceptés")
plt.scatter(x_ext, y_ext, color = 'red', label = "points rejetés")
traceCercle()
traceCarré(ax)
plt.axis('equal')
ax.set_title('simulation pour {} points tirés'.format(N))
plt.legend(loc = 'best', fontsize = 8)
plt.axis('off')
plt.savefig('simu1.svg')
plt.show()
```

Code python pour vérifier expérimentalement la valeur de π.

```
import numpy as np
import matplotlib.pyplot as plt

N = 1000
angle = np.linspace(0 , 2 * np.pi , 50)

def experiencePi(N) :
        x = np.random.uniform(-1,1,N)
        y = np.random.uniform(-1,1,N)
        x_ext = []
        x_int = []
        y_ext = []
        y_int = []
        for i in range (N) :
                if (x[i]*x[i] + y[i]*y[i]) <= 1 :
                        x_int.append(x[i])
                        y_int.append(y[i])
                else :
                        x_ext.append(x[i])
                        y_ext.append(y[i])
        return 4*(len(x_int) / len(x))
PiExperimental = [experiencePi(N) for i in range(1000)]
moyennePi = np.mean(PiExperimental)
ecarttypePi = np.std(PiExperimental)

fig , ax = plt.subplots()

plt.hist(PiExperimental, range = (2.9,3.4), bins = 100, color = 'green',
        label ="1000 tirages", edgecolor = 'black')
plt.text(2.9,55,"moyenne = {:.2f}".format(moyennePi))
plt.text(2.9,52,"écart-type = {:.2f}".format(ecarttypePi))
ax.set_xlabel("valeur de pi")
ax.set_ylabel("nombre de mesures")
plt.savefig("pi.svg")
plt.show()
```

Code python pour simuler une variable aléatoire de la fonction $f(x) = 6x(1-x)1_{(0\leq x\leq 1)}$.

```
import numpy as np
import matplotlib.pyplot as plt
N = 1000
x = np.linspace(0,1,N)
y = 6 * x * (1 - x)
a = 0
b = 1
M = 3/2 #max de la fonction
Px = a + (b - a) * np.random.uniform(0,1,N)
Py = M * np.random.uniform(0,1,N)
Pnt_statut = (Py <= 6 * Px * (1 - Px))
Pnt_accx = []
Pnt_accy = []
Pnt_rejx = []
Pnt_rejy = []
for i in range(N) :
        if Pnt_statut[i] ==True :
                Pnt_accx.append(Px[i])
                Pnt_accy.append(Py[i])
        else :
                Pnt_rejx.append(Px[i])
                Pnt_rejy.append(Py[i])

fig, ax = plt.subplots()
plt.scatter(Pnt_rejx, Pnt_rejy, color = 'red',label = 'point rejeté')
plt.scatter(Pnt_accx, Pnt_accy, color = 'green', label = 'point accepté')
plt.plot(x,y,color = 'black')
ax.set_title('Simulation par rejet pour {} point tirés'.format(N))
plt.legend(loc = 'upper left', fontsize = 8)
plt.savefig('algorejet.svg')

plt.show()
```