# ActivitéBernoulli1STMG

May 3, 2020

## 1 Activité découverte d'une épreuve de Bernoulli et simulations d'échantillons

En période de confinement, jouer à un jeu sur console est une occupation comme une autre. Malheureusement, Emma et Noan n'arrivent pas à se mettre d'accord sur le choix du jeu. La première serait davantage intéressée vers un jeu d'aventure alors que le deuxième lui voudrait mettre un jeu de combat. Pour faciliter leur décision, leurs parents les invitent à jouer au jeu du **PILE ou FACE**. *Emma choisit le côté PILE* et *Noan le côté FACE*.

**1°) Situation 0:** Ils lancent une pièce non truquée. * Quelle est la probabilité que le côté visible soit PILE ? On l'appellera la proportion théorique $p$. * Et FACE ?

Emma considère que Noan triche lorsqu'il lance la pièce et que cela influe sur le résultat. Aussi, leurs parents les convient à utiliser une solution programmée pour que l'humain n'intervienne plus dans le lancer. "Bonne idée" se disent-ils à ce moment-là...

**2°) Situation 1:** Voici une fonction en langage Python.

1. Exécuter-la en cliquant sur la flèche
   - Que vous renvoie-t-elle ?
2. Exécuter la plusieurs fois.
   - Est-ce toujours la même réponse ?
3. Comment définir le PILE et le FACE pour qu'il puisse répondre au besoin d'Emma et Noan ?

[1]:
```
from random import *

def piece():
  x=randint(0,1)        #randint(a,b) retourne aléatoirement un entier entre a␣
 ↪et b
  return x

print (piece())
```

```
0
```

**3°) Situation 2 :** Après reflexion, Emma et Noan estime qu'il faudrait avoir une majorité de piles ou de faces lorsqu'on lance la pièce un nombre de fois donné. Leurs parents leurs proposent ce programme.

[3]:
```
from random import *

def jeu1():
  S=0
  for i in range (5):      #range (5) signifie que i prend successivement les␣
 ↪valeurs de la liste [0,1,2,3,4]
    x=randint(0,1)         #randint(a,b) retourne aléatoirement un entier entre␣
 ↪a et b
    S=S+x
  return (S)

print (jeu1())
```

```
4
```

1. Combien de fois la pièce est-elle lancée ?
2. Que représente la variable S retournée par la fonction jeu1() ?
3. Est-ce toujours le même enfant qui gagne ?

**4°) Situation 3:** La fonction jeu1() a été modifiée. * Exécuter-le. * Interpréter le résultat dans le contexte de l'exercice.

[286]:
```
from random import *

def jeu2():
  S=0
  for i in range (10):
    x=randint (0,1)
    S=S+x
  return (S/10)

print(jeu2())
```

```
0.4
```

**5°)** On appelle épreuve de Bernoulli, une expérience aléatoire n'ayant que deux issues. L'une est considérée comme le Succès et l'autre comme l'Echec. Laquelle des situations 1, 2 ou 3 est une épreuve de Bernoulli?

**6°)** La probabilité d'avoir le Succès est le paramètre $p$ de cette épreuve. Quelle est la valeur du paramètre $p$ dans la situation 1 ?

**7°)** On définit la variable aléatoire X qui prend la valeur 1 en cas de succès et 0 en cas d'échec. Ainsi P(X=1) signifie obtenir un succès dans le jeu situation 1. * Que signifie P(X=0) ? * Quelle est sa valeur ?

**8°)** Déterminer alors la loi de probabilité de la situation 1 que l'on appellera Loi de Bernoulli.

**9°)** Déterminer l'espérance de la variable aléatoire X dans la situation 1.

**10°) Situation 4:** Simulation sur un échantillon

- Exécuter-le.
- Interpréter le résultat retourné par la fonction *echantillon1()*

[4]:
```
from random import *

def echantillon1():
  L=[]              #Création d'une liste vide
  for i in range (10):
    L.append(piece())      #Remplir la liste L
  return (L)

print (echantillon1())
```

```
[1, 0, 0, 0, 0, 0, 1, 0, 0, 0]
```

**11°) Histogramme d'une nouvelle situation :**

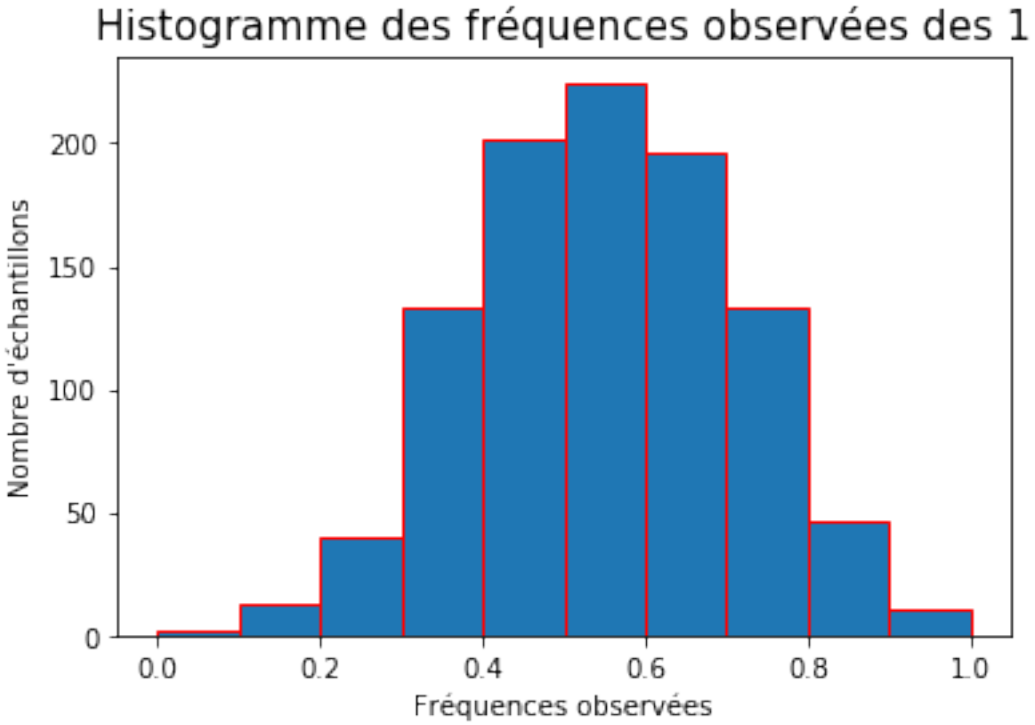
- Exécuter-le
- Modifier-le pour que vous obteniez une simulation fournissant les fréquences observées de 1 pour 1000 échantillons établit avec:
  - 50 lancers d'une pièce.
  - 100 lancers d'une pièce.
  - Interpréter les modifications de l'histogramme.

[5]:
```
from random import *
import matplotlib.pyplot as plt

def Proba_du_un(n):
  S=0
  for i in range (n):
    x=randint (0,1)
    S=S+x
  return (S/n)

#print (Proba_du_un(100))

def echantillon2():
  Liste=[]
  for i in range (1000):
    Liste.append(Proba_du_un(10))
  return (Liste)

print (echantillon2())

L=echantillon2()
plt.hist(L, bins=[0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1], edgecolor='red')
plt.title("Histogramme des fréquences observées des 1", fontsize=15)
plt.xlabel("Fréquences observées")
plt.ylabel("Nombre d'échantillons")
```

```
plt.show()
```

```
[0.4, 0.8, 0.6, 0.5, 0.3, 0.6, 0.7, 0.4, 0.5, 0.4, 0.5, 0.4, 0.5, 0.5, 0.4, 0.4,
0.5, 0.6, 0.8, 0.4, 0.6, 0.6, 0.6, 0.7, 0.4, 0.5, 0.4, 0.7, 0.5, 0.5, 0.5, 0.3,
0.3, 0.6, 0.4, 0.2, 0.6, 0.3, 0.5, 0.7, 0.6, 0.5, 0.5, 0.5, 0.2, 0.4, 0.4, 0.6,
0.5, 0.7, 0.2, 0.6, 0.3, 0.3, 0.3, 0.6, 0.4, 0.6, 0.5, 0.2, 0.5, 0.5, 0.6, 0.5,
0.3, 0.3, 0.4, 0.7, 0.3, 0.5, 0.4, 0.6, 0.5, 0.3, 0.5, 0.4, 0.8, 0.5, 0.5, 0.7,
0.7, 0.7, 0.7, 0.6, 0.4, 0.7, 0.4, 0.7, 0.5, 0.4, 0.6, 0.6, 0.7, 0.5, 0.4, 0.4,
0.7, 0.4, 0.6, 0.7, 0.3, 0.4, 0.6, 0.8, 0.3, 0.5, 0.6, 0.6, 0.6, 0.7, 0.3, 0.3,
0.7, 0.6, 0.4, 0.6, 0.4, 0.5, 0.5, 0.6, 0.4, 0.4, 0.6, 0.6, 0.5, 0.7, 0.5, 0.6,
0.3, 0.6, 0.4, 0.6, 0.7, 0.8, 0.3, 0.5, 0.6, 0.5, 0.5, 0.7, 0.2, 0.4, 0.6, 0.5,
0.3, 0.5, 0.2, 0.4, 0.6, 0.1, 0.6, 0.4, 0.3, 0.5, 0.6, 0.9, 0.4, 0.3, 0.4, 0.9,
0.4, 0.8, 0.5, 0.4, 0.5, 0.6, 0.5, 0.5, 0.4, 0.5, 0.7, 0.7, 0.4, 0.5, 0.5, 0.7,
0.1, 0.3, 0.5, 0.3, 0.2, 0.5, 0.2, 0.5, 0.6, 0.7, 0.3, 0.6, 0.4, 0.8, 0.2, 0.5,
0.2, 0.3, 0.2, 0.7, 0.5, 0.5, 0.4, 0.4, 0.4, 0.5, 0.5, 0.2, 0.7, 0.5, 0.7, 0.3,
0.6, 0.5, 0.5, 0.7, 0.4, 0.4, 0.7, 0.3, 0.4, 0.7, 0.3, 0.4, 0.4, 0.4, 0.7, 0.4,
0.4, 0.5, 0.5, 0.5, 0.5, 0.5, 0.7, 0.5, 0.4, 0.5, 0.4, 0.4, 0.7, 0.4, 0.4, 0.2,
0.7, 0.5, 0.7, 0.3, 0.5, 0.6, 0.5, 0.7, 0.6, 0.7, 0.4, 0.6, 0.5, 0.5, 0.3, 0.4,
0.5, 0.8, 0.5, 0.7, 0.5, 0.6, 0.4, 0.1, 0.6, 0.4, 0.6, 0.5, 0.4, 0.5, 0.3, 0.5,
0.6, 0.4, 0.6, 0.6, 0.6, 0.6, 0.5, 0.4, 0.3, 0.4, 0.6, 0.4, 0.5, 0.2, 0.5, 0.5,
0.5, 0.4, 0.3, 0.5, 0.4, 0.9, 0.4, 0.5, 0.6, 0.3, 0.5, 0.6, 0.6, 0.5, 0.7, 0.6,
0.5, 0.3, 0.5, 0.4, 0.3, 0.5, 0.5, 0.3, 0.6, 0.2, 0.4, 0.7, 0.4, 0.4, 0.3, 0.4,
0.6, 0.5, 0.6, 0.5, 0.5, 0.7, 0.4, 0.4, 0.4, 0.4, 0.3, 0.4, 0.5, 0.4, 0.5, 0.5,
0.5, 0.5, 0.7, 0.4, 0.2, 0.4, 0.5, 0.1, 0.4, 0.4, 0.8, 0.3, 0.4, 0.4, 0.2, 0.5,
0.6, 0.4, 0.5, 0.3, 0.3, 0.6, 0.8, 0.4, 0.3, 0.3, 0.7, 0.8, 0.6, 0.7, 0.4, 0.5,
0.7, 0.8, 0.6, 0.4, 0.5, 0.7, 0.7, 0.7, 0.5, 0.6, 0.5, 0.4, 0.4, 0.5, 0.7, 0.5,
0.3, 0.7, 0.6, 0.6, 0.3, 0.4, 0.6, 0.7, 0.6, 0.4, 0.6, 0.2, 0.6, 0.5, 0.6, 0.4,
0.5, 0.5, 0.4, 0.2, 0.5, 0.3, 0.6, 0.3, 0.5, 0.4, 0.4, 0.5, 0.4, 0.9, 0.6, 0.4,
0.4, 0.4, 0.3, 0.4, 0.8, 0.5, 0.3, 0.5, 0.6, 0.6, 0.5, 0.4, 0.4, 0.6, 0.7, 0.7,
0.4, 0.5, 0.4, 0.6, 0.4, 0.6, 0.3, 0.5, 0.5, 0.4, 0.5, 0.5, 0.3, 0.8, 0.3, 0.5,
0.4, 0.4, 0.4, 0.2, 0.5, 0.4, 0.3, 0.6, 0.6, 0.4, 0.6, 0.3, 0.7, 0.5, 0.4, 0.7,
0.1, 0.5, 0.4, 0.6, 0.4, 0.7, 0.5, 0.6, 0.4, 0.6, 0.6, 0.7, 0.4, 0.4, 0.7, 0.6,
0.5, 0.5, 0.4, 0.5, 0.5, 0.5, 0.5, 0.7, 0.6, 0.4, 0.4, 0.3, 0.3, 0.2, 0.5, 0.5,
0.6, 0.6, 0.3, 0.7, 0.5, 0.4, 0.4, 0.7, 0.7, 0.3, 0.3, 0.3, 0.6, 0.6, 0.6, 0.5,
0.8, 0.5, 0.5, 0.7, 0.4, 0.7, 0.8, 0.6, 0.7, 0.5, 0.2, 0.5, 0.3, 0.6, 0.5, 0.3,
0.4, 0.4, 0.6, 0.7, 0.3, 0.4, 0.3, 0.4, 0.4, 0.8, 0.6, 0.5, 0.4, 0.5, 0.4, 0.2,
0.7, 0.8, 0.7, 0.5, 0.7, 0.8, 0.8, 0.4, 0.6, 0.6, 0.4, 0.4, 0.5, 0.3, 0.6, 0.4,
0.6, 0.5, 0.4, 0.6, 0.3, 0.4, 0.6, 0.7, 0.5, 0.3, 0.5, 0.4, 0.3, 0.3, 0.6, 0.2,
0.6, 0.3, 0.6, 0.4, 0.5, 0.6, 0.5, 0.4, 0.6, 0.3, 0.3, 0.4, 0.4, 0.9, 0.6, 0.4,
0.5, 0.3, 0.6, 0.1, 0.6, 0.4, 0.5, 0.5, 0.4, 0.9, 0.3, 0.5, 0.2, 0.4, 0.6, 0.4,
0.6, 0.7, 0.8, 0.5, 0.3, 0.5, 0.5, 0.7, 0.9, 0.2, 0.5, 0.4, 0.5, 0.6, 0.3, 0.6,
0.4, 0.3, 0.5, 0.5, 0.4, 0.6, 0.6, 0.4, 0.4, 0.3, 0.3, 0.6, 0.5, 0.6, 0.3, 0.4,
0.4, 0.7, 0.3, 0.3, 0.4, 0.3, 0.5, 0.5, 0.3, 0.5, 0.5, 0.7, 0.5, 0.6, 0.5, 0.5,
0.8, 0.4, 0.7, 0.5, 0.7, 0.4, 0.7, 0.6, 0.5, 0.5, 0.4, 0.5, 0.8, 0.6, 0.6, 0.3,
0.4, 0.4, 0.4, 0.7, 0.6, 0.3, 0.6, 0.4, 0.5, 0.3, 0.4, 0.5, 0.5, 0.4, 0.6, 0.4,
0.6, 0.4, 0.5, 0.4, 0.5, 0.4, 0.5, 0.7, 0.6, 0.8, 0.4, 0.3, 0.3, 0.5, 0.5, 0.6,
0.5, 0.5, 0.5, 0.4, 0.1, 0.4, 0.3, 0.6, 0.4, 0.5, 0.5, 0.1, 0.6, 0.9, 0.4, 0.9,
```

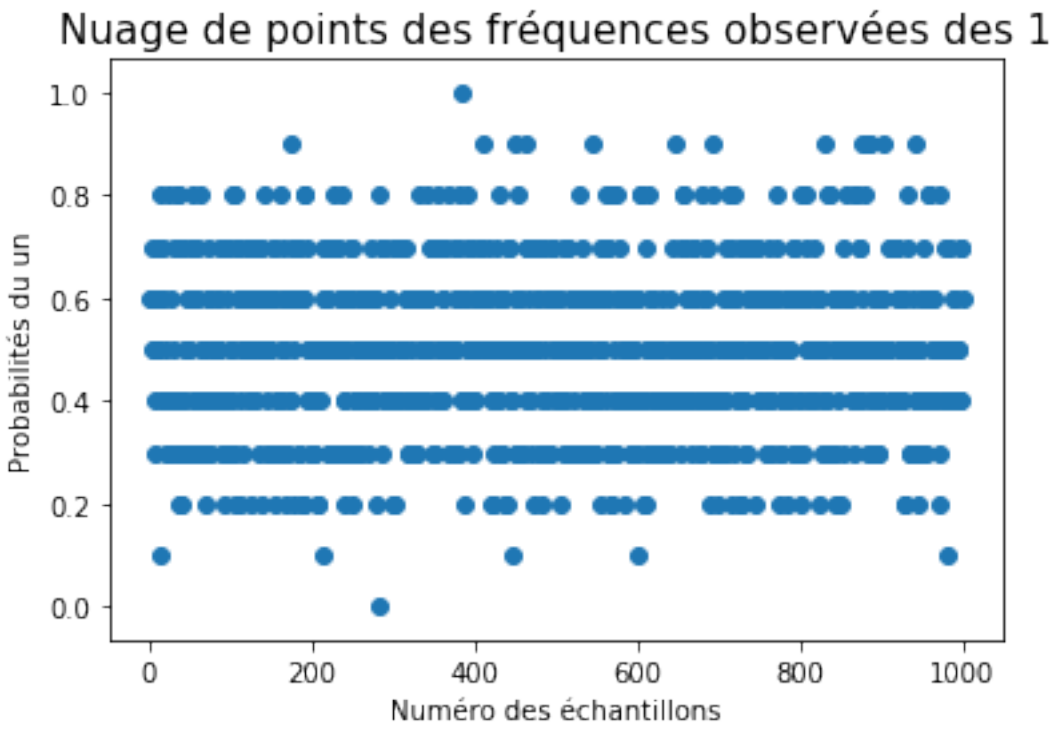
```
0.4, 0.5, 0.2, 0.7, 0.3, 0.5, 0.5, 0.4, 0.4, 0.4, 0.2, 0.2, 0.3, 0.5, 0.3, 0.7,
0.4, 0.5, 0.4, 0.4, 0.7, 0.7, 0.7, 0.6, 0.4, 0.5, 0.3, 0.7, 0.4, 0.3, 0.7, 0.7,
0.4, 0.5, 0.7, 0.5, 0.5, 0.7, 0.4, 0.3, 0.2, 0.4, 0.1, 0.4, 0.4, 0.2, 0.4, 0.4,
0.5, 0.4, 0.4, 0.5, 0.6, 0.6, 0.4, 0.7, 0.5, 0.4, 0.5, 0.4, 0.6, 0.8, 0.6, 0.7,
0.6, 0.5, 0.7, 0.4, 0.6, 0.3, 0.3, 0.5, 0.5, 0.6, 0.7, 0.5, 0.6, 0.7, 0.4, 0.5,
0.4, 0.5, 0.3, 0.5, 0.5, 0.5, 0.6, 0.4, 0.1, 0.7, 0.7, 0.4, 0.5, 0.6, 0.3, 0.7,
0.4, 0.3, 0.8, 0.6, 0.5, 0.4, 0.3, 0.7, 0.5, 0.2, 0.4, 0.4, 0.7, 0.7, 0.4, 0.6,
0.6, 0.5, 0.5, 0.6, 0.5, 0.4, 0.5, 0.4, 0.5, 0.6, 0.6, 0.2, 0.4, 0.6, 0.6, 0.4,
0.6, 0.1, 0.4, 0.5, 0.5, 0.4, 0.7, 0.7, 0.3, 0.3, 0.8, 0.3, 0.6, 0.5, 0.5, 0.3,
0.7, 0.7, 0.5, 0.6, 0.6, 0.5, 0.2, 0.4, 0.7, 0.4, 0.6, 0.6, 0.5, 0.3, 0.5, 0.5,
0.5, 0.4, 0.1, 0.4, 0.5, 0.4, 0.4, 0.6, 0.6, 0.6, 0.4, 0.4, 0.6, 0.6, 0.6, 0.7,
0.6, 0.5, 0.7, 0.6, 0.5, 0.2, 0.5, 0.4, 0.6, 0.3, 0.2, 0.4, 0.3, 0.8, 0.5, 0.3,
0.5, 0.4, 0.5, 0.4, 0.6, 0.7, 0.7, 0.7, 0.3, 0.5, 0.5, 0.5, 0.4, 0.3, 0.7, 0.4,
0.5, 0.3, 0.6, 0.5, 0.6, 0.1, 0.4, 0.5, 0.4, 0.2, 0.5, 0.3, 0.3, 0.4, 0.4, 0.8,
0.6, 0.4, 0.3, 0.4, 0.7, 0.5, 0.8, 0.6, 0.4, 0.3, 0.5, 0.5, 0.6, 0.4, 0.3, 0.7,
0.7, 0.4, 0.4, 0.5, 0.6, 0.4, 0.7, 0.2, 0.7, 0.5, 0.6, 0.4, 0.5, 0.5, 0.2, 0.2,
0.7, 0.5, 0.6, 0.6, 0.6, 0.3, 0.5, 0.5, 0.5, 0.6, 0.5, 0.2, 0.3, 0.5, 0.3, 0.6,
0.7, 0.7, 0.6, 0.5, 0.6, 0.6, 0.4, 0.7]
```




**12°) Nuage de points de la situation précédente:** * Exécuter-le. * Modifier-le pour que vous obteniez une simulation fournissant les fréquences observées de 1 pour 1000 échantillons établit avec: * 50 lancers d'une pièce. * 100 lancers d'une pièce. * Interpréter les modifications sur le nuage de points.

[41]:
```
from random import *
import statistics
import matplotlib.pyplot as plt

x=[]
for i in range(1,1001):
  x.append(i)

L=echantillon2()

ecarType=statistics.stdev(L)   #calcul de l'écart-type de la série des␣
 ↪fréquences observées de 1
print("L'écart-type de la série est égale à",round(ecarType,3)) #Arrondir à␣
 ↪0,001 près.

plt.scatter(x,L)
plt.title("Nuage de points des fréquences observées des 1", fontsize=15)
plt.xlabel("Numéro des échantillons")
plt.ylabel("Probabilités du un")
plt.show()
```

```
L'écart-type de la série est égale à 0.163
```

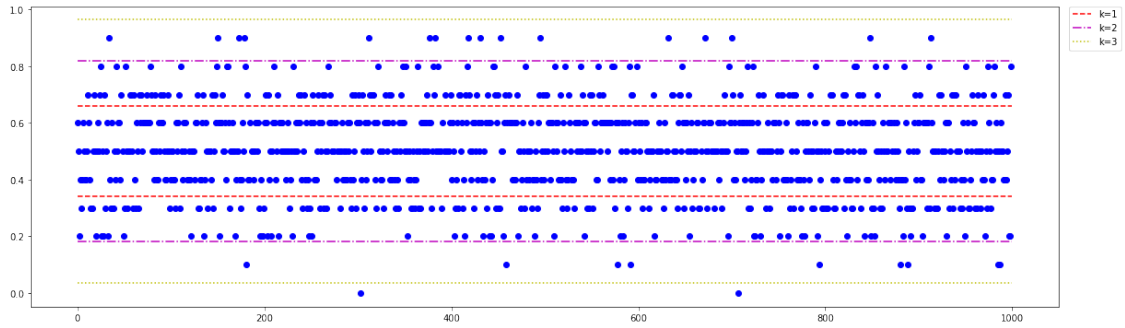



### 13°) Interprétation des résultats

On note $s$ l'écart-type et $p$ la proportion théorique d'obtention du PILE dans le lancer d'une pièce.

[26]:
```
from random import *
import statistics
fig, (ax1) = plt.subplots(1, figsize=(20, 6))

def bornes(L,k):  #Détermine les bornes de l'intervalle d'étude
    p = 0.5
    s =statistics.stdev(L) #calcul de l'écart-type de la série des fréquences␣
 ↪observées de 1
    return [round(p-k*s,3),round(p+k*s,3)]

def propPoints(L,k):   #Détermine la proportions de points situés dans␣
 ↪l'intervalle d'étude
    b,B = bornes(L,k)
    filtre = [x for x in L if x<=B and x>=b]
    return len(filtre)/len(L)

for k in range (1,4):
    print ("L'intervalle [p-{0}*s,p+{0}*s] est : ".format(k),␣
 ↪bornes(echantillon2(),k))
    print ("La proportion de points entre [p-{0}*s,p+{0}*s] est ".
 ↪format(k),propPoints(echantillon2(),k))

i1,s1 =  bornes(echantillon2(),1)
i2,s2 =  bornes(echantillon2(),2)
i3,s3 =  bornes(echantillon2(),3)

ax1.plot(echantillon2(),'bo')
ax1.plot([0,1000],[i1,i1],'r--',label='k=1')
ax1.plot([0,1000],[s1,s1],'r--')
ax1.plot([0,1000],[i2,i2],'m-.',label='k=2')
ax1.plot([0,1000],[s2,s2],'m-.')
ax1.plot([0,1000],[i3,i3],'y:',label='k=3')
ax1.plot([0,1000],[s3,s3],'y:')
ax1.legend(bbox_to_anchor=(1.01, 1), loc=2, borderaxespad=0.)
plt.show()
```

```
L'intervalle [p-1*s,p+1*s] est :  [0.34, 0.66]
La proportion de points entre [p-1*s,p+1*s] est  0.659
L'intervalle [p-2*s,p+2*s] est :  [0.172, 0.828]
La proportion de points entre [p-2*s,p+2*s] est  0.98
L'intervalle [p-3*s,p+3*s] est :  [0.025, 0.975]
La proportion de points entre [p-3*s,p+3*s] est  0.998
```


1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
k=1
k=2
k=3

[ ]: