M1 Physique Appliquée
Année 2016-2017

Université Paris-Sud XI

Programmation et données numériques

TD : Incertitudes et barres d'erreur

Création d'un module ou librairie

On va maintenant utiliser cela pour créer un module, ou librairie, que l'on appelle `Statistique` et qui doit par conséquent être écrit dans un fichier appelé `Statistique.py`

☞ Créer un fichier `Statistique.py`. Y inclure les définitions suivantes que l'on complétera, `d` étant une liste. On importe `numpy` et `pyplot` afin de disposer par la suite de toutes les fonctions mathématiques ainsi que des autres fonctions utiles de numpy.

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

def moyenne(d):
    if len(d): return  ........
    else:      return 0.0

def variance(d):
    if len(d) < 2: return 0.0
    moy =  ......
    return sum([  .......  for x in d ])/(len(d)-1.0)
```

☞ Vous pouvez alors importer les fonctions de cette librairie ou bien la librairie en entier, ou bien la librairie renommée

```
from Statistique import moyenne, variance
from Statistique import *
import Statistique as stat
```

Pour vérifier que tout marche bien, écrire un script `scriptPlot.py` qui affiche un plot selon

```
from Statistique import *

x = np.linspace(0,10,101)
y = np.sqrt(x)
plt.plot(x,y)
plt.show()
```

☞ Faites un essai de l'utilisation de ces fonctions dans un script avec une liste de votre choix en utilisant la ligne de commande.

Une barre d'erreur statistique

On illustre ici certains points discutés en cours sur une seule barre d'erreur. On écrira un script `BarreErreur.py`.

☞ Pour commencer, créer une fonction qui va simuler une série de N mesures distribuées selon une loi uniforme

```
from Statistique import moyenne, variance, plt
from random import uniform

mu, dispersion = 3.245, 0.234
vmin, vmax = mu-dispersion/2., mu+dispersion/2.
def Serie(N):
    return        ...........................
print Serie(3)
```

Compléter la ligne ci-dessus. Quelle est la variance exacte, à entrer dans une variable `exactvar`, de cette loi en fonction de `dispersion` ?

☞ On veut tracer la distribution de la moyenne obtenue pour différents nombre de mesures. Compléter le code ci-dessous et commenter le graphe :

```
Nmesures = [1,2,3,5,10,24,50,120,300]
Stat = 20000
res = []
for N in Nmesures:
    res.append (....... Serie(N  )........)

i = 0
for r in res [::2]:
    i += 1
    n, bin, patches = plt.hist(r, bins=51, range=(vmin,vmax), \
                                normed=1, color=str(i/float(len(res))) )
plt.xlim  ([.....,....])
plt.xlabel("estimation de la moyenne")
plt.ylabel(u"densité de probabilité")
plt.tight_layout()
```

Rajouter une ligne pour faire afficher une légende sur le graphe.

☞ Observons maintenant comment la barre d'erreur diminue avec le nombre de mesures N. Compléter le script ci-dessous pour obtenir le graphe montrant les moyennes des résultats pour différents N avec une barre d'erreur correspondant à un écart-type. Quel est le comportement attendu ? Modifier les deux lignes `plt.plot(....)` pour tracer ce comportement exact en fonction de N

```
y = [  .......  for r in res ]
sigma_y = [  .......  for r in res ]

ns = np.logspace(0,3,100)
plt.plot ([0.5,500],[ mu,mu],'k--')
plt.plot  (..........)
plt.plot  (..........)
plt.errorbar(Nmesures, y, yerr=sigma_y, fmt='o', color='b', capthick=2, capsize=6)
plt.xscale('log')
plt.xlim ([0.5,500]);  plt.ylim([vmin,vmax])
plt.xlabel("nombre de mesures")
plt.ylabel(u"valeur estimée avec barre d\'erreur")
```

Formule de propagation des erreurs

On rappelle que si deux variables aléatoires X et Y sont reliées par une loi du type $y = f(x)$, on peut estimer le lien entre les écart-types de la manière suivante

$$\sigma_y^{prop} \simeq \left|\frac{df}{dx}\right| \sigma_x \tag{1}$$

☞ On considère une relation linéaire de la forme $y = ax + b$. Que peut-on dire de la formule de propagation des erreurs dans ce cas ? Comprendre puis compléter le script ci-dessous qui génère des données bruitées $\{x_k, y_k\}$ et qui réutilise la bibliothèque `Statistique.py`.

```
from Statistique import variance, np, plt
from random import gauss

sigma = 0.1

modele = lambda x: x + 1.0
sigma_prop = lambda x: sigma

def data_y(x,Nstat=10000):
    y = []
    for k in range(Nstat):
        xk = x + gauss(0.0,sigma)
        y.append (......)
    return y

plt.hist(data_y(0.1),bins=51)
plt.hist(data_y(0.5),bins=51)
plt.show()

Npoints = 51
xk = np.linspace(0.1, 3.0, Npoints)
ratio = []
for x in xk:
    res = np.sqrt(variance(data_y(x)))
    ratio.append(res/sigma_prop(x))

txt = "Test de la formule de propagation des erreurs"
plt.plot(xk,ratio,'ko',linewidth=3)
plt.plot(xk,[1.0]*len(xk),'r-',linewidth=2)
plt.xlim([min(xk)-0.5,max(xk)+0.5]);
plt.title(txt)
plt.xlabel("x"); plt.ylabel("$\sigma_y/\sigma_y^{prop}$")
plt.tight_layout()
plt.show()
```

Observer le comportement du rapport σ_y/σ_y^{prop} pour des σ_x de plus en plus grands.

☞ Modifier le script ci-dessus pour étudier la formule de propagation des erreurs pour un modèle de type $y = x^q$ avec q donné, typiquement un entier $q = 2, 3, \ldots$. Observer le comportement du rapport σ_y/σ_y^{prop} pour des `sigma` de plus en plus grands ainsi que le comportement des deux distributions tracées au début. Quel argument qualitatif sur la moyenne de y et son écart-type donne lieu à un bon accord avec la formule attendue. En déduire une interprétation pour les valeurs de x pour lesquelles le rapport devient moins bon.

☞ Reproduire la discussion pour le modèle $y = e^x$. Quand la formule de propagation des erreurs est-elle raisonnable dans ce cas ?

Autour du coefficient de corrélation linéaire

On rappelle la définition du coefficient de corrélation linéaire r pour une série de mesure de couples (x_k, y_k) :

$$r = \frac{\sum_k (x_k - \bar{x})(y_k - \bar{y})}{\sqrt{\sum_k (x_k - \bar{x})^2 \sum_k (y_k - \bar{y})^2}} \tag{2}$$

On souhaite écrire une fonction qui effectue ce calcul, puis l'utiliser pour illustrer son comportement.

☞ Quelles sont les valeurs possibles de r si les points (x_k, y_k) sont alignés selon une droite ?

☞ Compléter la fonction ci-dessous à inclure dans le fichier `Statistique.py`

```
def CoefficientCorrelation(x,y):
    if len(x)!=len(y) or not len(x):
        print "Error"; return None
    Ks = range(len(x))
    xbar, ybar =     ................
    sigma_xy =     ...................
    sigma_x  =     ...................
    sigma_y  =     ...................
    return sigma_xy/np.sqrt(sigma_x*sigma_y)
```

☞ Traçons un exemple de données construites selon les lignes dans un script `Correlations.py` :

```
from Statistique import CoefficientCorrelation, plt
from random import uniform

N = 10
x, y = [ uniform(0,1) for i in range(N) ],[ uniform(0,1) for i in range(N) ]
r = CoefficientCorrelation(x,y)
plt.plot(x,y,'o')
plt.title("$r= {:.5f}$".format(r))
plt.tight_layout()
plt.show()
```

Les données `x` et `y` sont-elles indépendantes ? Calculer leur coefficient de corrélations pour quelques nombres d'échantillons N différents.

☞ On souhaite effectuer une étude plus systématique en traçant la distribution de r pour différents N

```
N = 10
data = []
for stat in range(20000):
        ......................
n, bin, patches = plt.hist(data, bins=21, range=(-1.0,1.0), normed=1)
plt.show()
```

Compléter la ou les lignes manquantes dans la boucle. Discutez ce que vous observez pour $N = 2, 3, 4$ puis augmentez, raisonnablement N jusqu'à 100.

☞ On se place dans le cas suivant :

```
theta = [ uniform(0,2*np.pi) for i in range(N) ]
x, y = np.cos(theta), np.sin(theta)
```

Les variables x et y sont-elles indépendantes ? Calculer quelques coefficients de corrélations. Effectuer de nouveau l'étude systématique de la distribution de r et discuter les résultats.

☞ Enfin, pour des exemples tirés de la vie réelle de corrélations douteuses, on pourra consulter le site `http://www.tylervigen.com/spurious-correlations`.