

Ouverture vers le supérieur

loi normale · régression · Monte-Carlo · scipy · tests · documentation

LOI NORMALE · ATTENDU DE PCSI

N tirages gaussiens	<code>np.random.normal(m, s, N)</code>
$[m - s ; m + s]$	$\approx 68\%$ des valeurs
$[m - 2s ; m + 2s]$	$\approx 95\%$: intervalle de confiance
proportion	<code>np.mean(abs(x - m) < s)</code>
histogramme normé	<code>plt.hist(x, bins=40, density=True)</code>
exponentielle	<code>np.exp(x)</code>

COMPARER, VALIDER · CPGÉ

écart normalisé	$ x_1 - x_2 / \sqrt{u_1^2 + u_2^2}$
seuils (cours)	< 1 très compatibles · ≥ 2 non compatibles
barres d'incertitude	<code>plt.errorbar(x, y, yerr=u, fmt='o')</code>
écarts au modèle	$E = (y - (a \cdot x + b)) / u$
modèle validé si	tous $ E < 2$, signes au hasard
R^2 seul	ne valide rien

TESTER · INFORMATIQUE DE CPGÉ

spécification	<code>"""Ce que reçoit, ce que renvoie."""</code>
test	<code>assert abs(f(2) - 0.5) < 1e-9</code>
dictionnaire	<code>d = {'NH4+' : 9.2} · d['NH4+']</code>
ajouter, chercher	<code>d['cle'] = v · 'cle' in d</code>
parcourir	<code>for k, v in d.items():</code>

SCIPY · ATTENDU DE PCSI

racine entre a et b	<code>from scipy.optimize import bisect</code> <code>bisect(f, a, b)</code> , f(a) et f(b) de signes contraires
Newton (PCSI chimie)	$x \leftarrow x - g(x) / g'(x)$ <code>newton(g, x0, fprime=gp)</code>
équation différentielle	<code>from scipy.integrate import odeint</code> <code>sol = odeint(derivee, y0, t)</code>
ordre de derivee	<code>def derivee(y, t):</code> y d'abord
ordre 2	$Y = [\theta, \omega] \rightarrow$ renvoyer $[\omega, d\omega/dt]$
une colonne	<code>sol[:, 0]</code>

MONTE-CARLO SUR UNE RÉGRESSION

liste vide, ajout	<code>pentres = [] · pentres.append(a)</code>
mesures simulées	<code>y + np.random.normal(0, u, len(y))</code>
u(pente)	<code>np.std(pentres, ddof=1)</code>

DOCUMENTATION MATPLOTLIB

épaisseur, style	<code>linewidth=3 · linestyle='--'</code>
couleur nommée	<code>color='darkorange'</code>
droite verticale	<code>plt.axvline(x) · plt.axhline(y)</code>
échelle log	<code>plt.yscale('log')</code>
2 graphiques empilés	<code>plt.subplot(2, 1, k)</code>
flèche et texte	<code>plt.annotate('...', xy=..., xytext=..., arrowprops={'arrowstyle': '->'})</code>

EXEMPLE COMMENTÉ 1 · L'ÉTAT FINAL D'UN ACIDE FAIBLE AVEC BISECT

```
import numpy as np
from scipy.optimize import bisect      # 14 Mo téléchargés la 1re fois

Ka = 10 ** (-4.8)                      # acide éthanoïque
C = 1.0e-2                             # concentration apportée, en mol/L

def f(x):
    """Écart à l'équilibre ; x = [H3O+] en mol/L, 0 <= x < C.""" # la spécification
    return x**2 / (C - x) - Ka

assert f(0) < 0 and f(0.001) > 0      # on vérifie le changement de signe AVANT
x_f = bisect(f, 0, 0.001)              # f sans parenthèses : on passe la fonction
print('tau =', round(x_f / C, 4), ' pH =', round(-np.log10(x_f), 2))
```

```
tau = 0.039  pH = 3.41
```

Ne pas prendre $b = C : C - x$ s'annulerait (division par zéro).

EXEMPLE COMMENTÉ 2 • VALIDER UNE DROITE, PUIS L'INCERTITUDE SUR SA PENTE

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

c = np.array([1.0, 2.5, 5.0, 8.0, 10.0])      # TP4, en mmol/L
sigma = np.array([0.162, 0.330, 0.605, 0.925, 1.158]) # en mS/cm
u_sigma = 0.015                               # incertitude-type sur chaque  $\sigma$ 

# 1. régression, puis écart normalisé de chaque point au modèle
a, b = np.polyfit(c, sigma, 1)
E = (sigma - (a * c + b)) / u_sigma           # tous |E| < 2 et signes mêlés : modèle validé
print('E =', np.round(E, 2))                  # np.round arrondit tout un tableau

# 2. Monte-Carlo : on refait la régression sur 1000 jeux de mesures simulés
np.random.seed(1)                             # fixe le hasard : mêmes tirages à chaque exécution
pentes = []
for i in range(1000):
    sigma_sim = sigma + np.random.normal(0, u_sigma, 5) # chaque point bouge au hasard
    p, o = np.polyfit(c, sigma_sim, 1)
    pentes.append(p)                             # on garde la pente de ce tirage
print('a =', round(a, 4), ' u(a) =', round(np.std(pentes, ddof=1), 4))

# 3. le graphique « raisonné » : barres d'incertitude + modèle
plt.errorbar(c, sigma, yerr=u_sigma, fmt='o', label='mesures')
plt.plot(c, a * c + b, 'r-', label='modèle affine')
plt.legend()
plt.show()
```

```
E = [-0.07  0.13  0.13 -0.53  0.33]
a = 0.11  u(a) = 0.0019
```

Résultat : $\alpha = (0,110 \pm 0,002) \text{ mS} \cdot \text{cm}^{-1} \text{ par mmol} \cdot \text{L}^{-1}$. Python écrit 0.11 : il n'affiche pas les zéros de fin.

EXEMPLE COMMENTÉ 3 • UN PENDULE AVEC ODEINT

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

g = 9.81
L = 1.0

def derivee(Y, t):
    # Y d'abord, t ensuite : imposé par odeint
    # ordre 2 → système de 2 équations d'ordre 1
    theta, omega = Y
    return [omega, -g / L * np.sin(theta)]

t = np.linspace(0, 10, 2001)
for deg in [10, 90]:
    sol = odeint(derivee, [deg * np.pi / 180, 0], t) #  $\vartheta(0)$  en rad,  $\omega(0) = 0$ 
    plt.plot(t, sol[:, 0] * 180 / np.pi, label=str(deg) + '°') # colonne 0 :  $\vartheta$ 

print('T0 =', round(2 * np.pi * np.sqrt(L / g), 2), 's')
plt.xlabel('t (s)')
plt.ylabel('θ (°)')
plt.legend()
plt.show()
```

```
T0 = 2.01 s
```

Sur le graphique : $T \approx 2,0 \text{ s}$ à 10° , mais $\approx 2,4 \text{ s}$ à 90° . Les grandes oscillations ne sont pas isochrones.

LES PIÈGES

écarts normalisés	<code>/ u</code>	Sans division par l'incertitude, ce sont des écarts bruts : le seuil 2 n'a plus de sens.
hist + Gauss	<code>density=True</code>	Sans lui, l'histogramme compte des effectifs et la courbe de Gauss paraît plate.
bisect	signes contraires	<code>ValueError</code> si $f(a)$ et $f(b)$ ont le même signe : regarder d'abord le graphique.
bisect(f, ...)	<code>f</code>	On passe la fonction, sans parenthèses ; <code>f(x)</code> serait un nombre.
derivee	<code>(y, t)</code>	odeint appelle la fonction avec y d'abord ; l'ordre inverse donne un résultat faux, sans message.
sin θ	<code>np.sin(theta)</code>	Remplacer sin θ par θ n'est valable que pour les petits angles.
flottants	<code>abs(a - b) < 1e-9</code>	<code>abs(2.05 - 2.10) / 0.10</code> vaut <code>0.50000000000000027</code> : jamais de <code>==</code> entre deux nombres à virgule.
if / elif	le plus petit seuil d'abord	Les conditions sont lues dans l'ordre : la première vraie l'emporte.
Monte-Carlo	<code>y + tirage</code>	Tirer le bruit seul, sans l'ajouter aux mesures, donne des pentes nulles.
$R^2 = 0,999$	écarts normalisés	Des points presque alignés peuvent former une courbe : seuls les écarts normalisés le révèlent.

D'OÙ VIENNENT CES ATTENDUS

CPGE, toutes filières de 1 ^{re} année	loi normale et intervalle de confiance, écart normalisé, régression validée, Monte-Carlo sur les paramètres (bloc « Mesures et incertitudes »)
PCSI, outils numériques	<code>random.normal</code> , <code>hist</code> , <code>polyfit</code> , <code>bisect</code> , <code>odeint</code> ; Newton et état final avec K en chimie
informatique commune de CPGE	spécification, jeux de tests, <code>assert</code> , dictionnaires, précision des flottants
BUT Mesures physiques	test d'un programme (R1.07), représentations graphiques des mesures (SAÉ 1.01)
BUT Chimie	statistiques descriptives et régression linéaire en TP (R1.07)
BTS Métiers de la chimie	incertitude élargie et niveau de confiance

CHERCHER DANS LA DOCUMENTATION

Une page de fonction se lit toujours ainsi : la **signature**, puis **Parameters** (chaque paramètre, ce qu'il attend), **Returns**, et des **Examples** à recopier. Point de départ : matplotlib.org/stable/api/pyplot_summary.html. Vocabulaire : *width* largeur · *dashed* tirets · *scale* échelle · *arrow* flèche · *row* ligne.

CE QUE JE SAIS FAIRE

- ☐ Simuler une loi normale et associer un intervalle de confiance.
- ☐ Superposer une courbe de Gauss à un histogramme normé.
- ☐ Comparer deux mesures par leur écart normalisé.
- ☐ Valider une régression : barres d'incertitude, écarts normalisés.
- ☐ Évaluer par Monte-Carlo l'incertitude sur une pente.
- ☐ Résoudre une équation avec `bisect` ou Newton.
- ☐ Résoudre une équation différentielle avec `odeint`.
- ☐ Spécifier, tester, utiliser un dictionnaire.
- ☐ Trouver un paramètre dans la documentation.

Point à revoir :

.....

.....