Python#005★★☆☆ 2022.4.21 / 2024.4.10

薬物動態～経口投与反復 1-コンのあてはめ計算

薬物動態、経口反復投与 1-コンパートメントモデルのあてはめ計算（prog02_12.py）

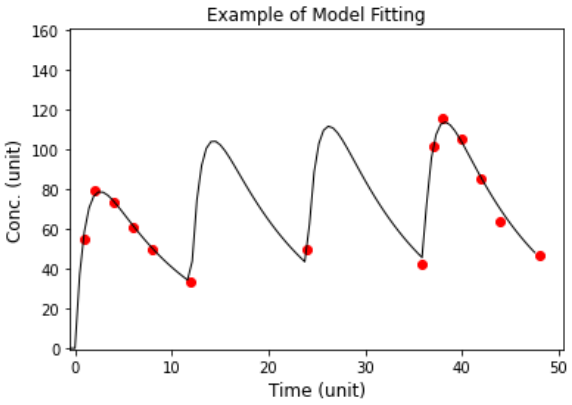
＊前回までの記事を確認した上で下記を参照のこと。

(1) 投与量（DOSE)、薬物動態パラメータ（クリアランス（CL)、分布容積（VD)、吸収速度定数（KA))。反復投与は一定用量・一定間隔（TAU）とした。

(2) minimize 関数を用いた。残差平方和を関数 obj_func() に、モデル式を model_eq121() に定義。

(3) 薬物動態パラメータはリスト（P[]）とし、グローバル変数とした。

以上の内容で作成した例を次ページ「prog02_12.py」に示した。出力結果は次のとおり。




```
-- Estimated Parameters --
     CL = 0.1014
     Vd = 0.9647
     Ka = 0.9432
     Ke = 0.1051
 * t1/2 = 6.5957
```

```
 final_simplex: (array([[0.10138255, 0.96471875, 0.94317568],
       [0.10137955, 0.96474343, 0.9432075 ],
       [0.10138128, 0.96465381, 0.94311458],
       [0.10137819, 0.96471284, 0.94310344]]), array([125.69856873, 125.69857812,
125.69860529, 125.69861858]))
           fun: 125.69856872941324
       message: 'Optimization terminated successfully.'
          nfev: 98
           nit: 52
        status: 0
       success: True
             x: array([0.10138255, 0.96471875, 0.94317568])
```

以上

```
"""# Program 02-12 "prog02_12.py"""
# Fitting to One-Compartment po model after multiple dosing
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as opt

# Global varialbes
DOSE = 100.
TAU = 12.0 # Dosing Interval

Nparam = 3
P = [0 for j in range(Nparam)]
P[0] = 0.1 # CL
P[1] = 1.0 # Vd (Vd/F)
P[2] = 1.5 # Ka
Npoint = 14
x = [0 for j in range(Npoint)]
y = [0 for j in range(Npoint)]

#
def set_data():
    # Data input in this program
    data_x = np.array([1.0, 2.0, 4.0, 6.0, 8.0, 11.9, 23.9, 35.9,
             37, 38, 40.0, 42.0, 44.0, 48.0])
    data_y = np.array([55.1, 79.2, 73.5, 60.7, 49.5, 33.2, 49.5, 42.1,
             101.2, 115.3, 105.2, 85.3, 63.9, 46.8])
    return data_x, data_y

def obj_func(P):
    SS = 0.0
    for i in range(Npoint):
        xx = x[i]
        Cp = model_eq121(xx, i, P)
        SS = SS + (y[i] - Cp) ** 2
    return SS

def model_eq121(xx, i, P):
    # 1-Comp. po model (no lag time)
```

```
    tn = 0
    x2 = 0
    while xx > tn * TAU:
        tn = tn + 1
        x2 = xx - (tn - 1) * TAU
    CL = P[0]
    VD = P[1]
    ka = P[2]
    ke = CL / VD
    E1 = 1 - np.exp(-ke * tn * TAU)
    E2 = 1 - np.exp(-ke * TAU)
    E3 = 1 - np.exp(-ka * tn * TAU)
    E4 = 1 - np.exp(-ka * TAU)
    E5 = DOSE / VD * ka / (ka - ke)
    Cp = E5 * (E1 / E2 * np.exp(-ke * x2) - E3 / E4 * np.exp(-ka * x2))
    return Cp

def draw_plot_obs(data_x, data_y):
    # Plotting observed data
    plt.title("Example of Model Fitting", fontsize = 12)
    plt.xlim(-0.5, 50.5)
    plt.ylim(-0.5, 160.5)
    plt.xlabel("Time (unit)", fontsize = 12)
    plt.ylabel("Conc. (unit)", fontsize = 12)
    plt.plot(data_x, data_y, "o", color = "red")
    #plt.show()

def draw_plot_pred():
    # Simulation of predicted curve
    tx = [0 for j in range(100)]
    cy = [0 for j in range(100)]
    for i in range(100):
        tx[i] = (i - 1) * (48.0 - 0.0) / (100 - 1)
        cy[i] = model_eq121(tx[i], i, P)
    plt.plot(tx, cy, linestyle = "solid", linewidth = "1", color = "black")
    plt.show()

def print_output():
```

```
    # Output
    print ("-- Estimated Parameters --")
    print ('    CL = {:.4f}'.format(CL))
    print ('    Vd = {:.4f}'.format(VD))
    print ('    Ka = {:.4f}'.format(KA))
    print ('    Ke = {:.4f}'.format(CL / VD))
    print (' * t1/2 = {:.4f}'.format(np.log(2) / (CL / VD)))

#
if __name__ == '__main__':
    x, y = set_data()
    draw_plot_obs(x, y)
    results = opt.minimize(obj_func, P, method='Nelder-Mead')
    print (results)
    P[0] = results.x[0]
    P[1] = results.x[1]
    P[2] = results.x[2]
    CL = P[0]
    VD = P[1]
    KA = P[2]
    draw_plot_pred()
    print_output()
```