Python#001★☆☆☆ 2022.1.26 / 2023.7.21 / 2024.4.8

薬物動態：静脈内瞬時投与 1-コンパートメントモデルのあてはめ計算

薬物動態、静脈内瞬時投与 1-コンパートメントモデルのあてはめ計算（prog02_01.py）

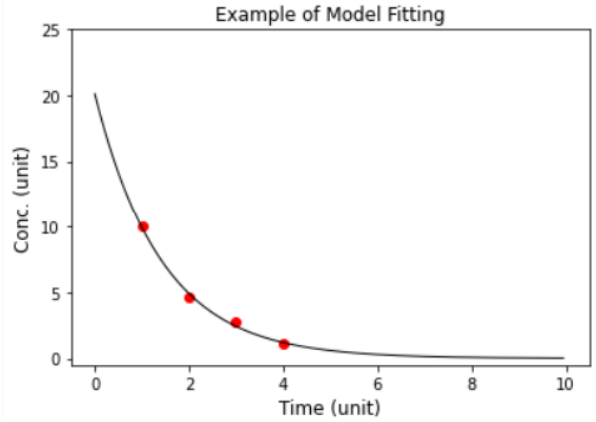
(1) データの入力：データ解析を行う際には測定データを設定することから始まる。血中濃度等の測定データを EXCEL に整理し、そこから python でデータを読み込むことも可能であるが、ここでは簡潔にプログラムの中にデータを記載してみた：data_x = [1.0, 2.0, 3.0, 4.0]、data_y = [10.0, 4.6, 2.8, 1.1]（x が測定時間、y が測定値に相当する）。投与量（DOSE）、薬物動態パラメータであるクリアランス（CL）、分布容積（VD）はグローバル変数とした。

(2) モデルあてはめ：最小二乗法となる。ここでは python の Scipy から curve_fit をインポートして用いた。かつては最小二乗法アルゴリズムをもとにその計算ルーチンをプログラミングする必要があったが、今では不要である。モデルは関数（def）として定義し、今後モデル式を変更する場合に柔軟に対応できるようにした。

(3) 結果のグラフ化：実測値のプロットとあてはめ計算で得られた理論値（シミュレーション）をグラフに描画する。Matplotlib を用いた。

(4) 得られたパラメータ値の出力：format 機能を用いて整頓して表示できるようにしてみた。二次的パラメータとして ke、t1/2（半減期）も算出した。

以上の内容で作成した例を次ページ「prog02_01.py」にテキストとして示した。出力結果は次のとおり。




```
-- Estimated Parameters --
     CL = 3.5087
     Vd = 4.9724
     Ke = 0.7056
 * t1/2 = 0.9823
```

以上

```
"""# Program 02-01 "prog02_01.py"""
# Fitting to One-Compartment iv model
#
#import math
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
#
# Global varialbes
DOSE = 100.
CL = 1.
VD = 1.
#
def set_data():
    # Data input in this program
    data_x = [1.0, 2.0, 3.0, 4.0]
    data_y = [10.0, 4.6, 2.8, 1.1]
    return data_x, data_y
#
def model_eq111(x, CL, VD):
    # 1-Comp. iv model
    return DOSE / VD * np.exp( -CL / VD * x)
#
def draw_plot(data_x, data_y):
    # Data plotting
    plt.title("Example of Model Fitting", fontsize = 12)
    plt.xlim(-0.5, 10.5)
    plt.ylim(-0.5, 25.0)
    plt.xlabel("Time (unit)", fontsize = 12)
    plt.ylabel("Conc. (unit)", fontsize = 12)
    plt.plot(data_x, data_y, "o", color = "red")
    # Simulation of predicted curve
    pred_x = np.arange(0., 10., 0.05)
    pred_y = model_eq111(pred_x, CL, VD)
    #print (pred_x)
    #print (pred_y)

    plt.plot(pred_x, pred_y, linestyle = "solid", linewidth = "1", color = "black")
```

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

if __name__ == '__main__':
    x, y = set_data()
    p_init = (4., 4.)
    param, cov = curve_fit(model_eq111, x, y, p0 = p_init)
    CL = param[0]
    VD = param[1]
    draw_plot(x, y)
    print_output()
```