Python#007★★☆☆ 2022.5.2/ 2022.5.5

薬物動態～Sawchuk Zaske 法

静脈内点滴の薬物の体内動態を評価する Sawchuk-Zaske 法は、体内動態に 1-コンパートメントモデルを仮定して比較的簡便な計算で薬物動態パラメータを求め、さらに目標ピーク濃度、トラフ濃度が得られるような推奨投与方法を計算する。数式等の詳細は別記事で記載した。ここではプログラムとその実行結果についてのみ示す。

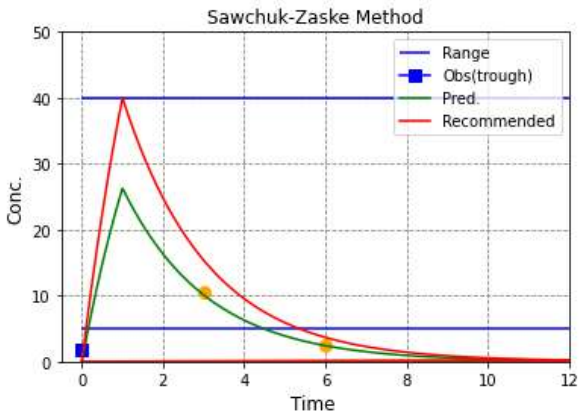
次のデータ（左）を使用した。結果を右に示す（目標 Cmax を 40μg/mL、Cmin を 5.0μg/mL とした）。

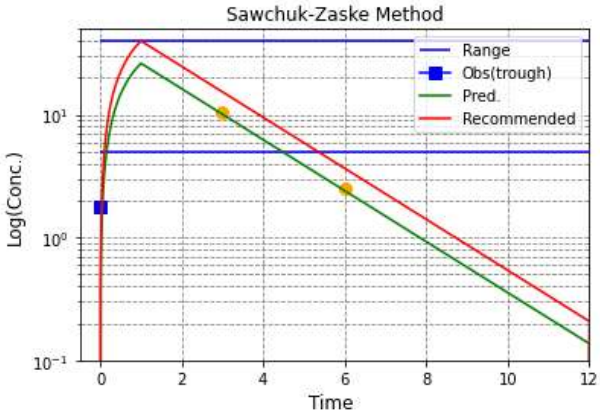
```
== Sawchuk-Zaske Method ==
-- Dosing Condition --
 Inf. Rate = 120.0
 Inf. Time = 1.0
 Interval. = 12.0
-- Obserbed Data --
 Trough = 1.80
 1-th data
  Time  = 3.0
  Conc. = 10.50
 2-th data
  Time  = 6.0
  Conc. = 2.50
```

```
-- Estimated Parameters --
  Ke = 0.478
  Vd = 3.638
  CL = 1.740
-- Recommendation --
  Interval  = 5.35
  Inf. Rate = 168.90
  * target peak   = 40.0
  * target trough = 5.0
```

結果の図を示す（左：通常スケール、右：片対数スケール）。■がトラフ実測値、●が消失相での実測値を示す。曲線のうち下側が実測値から求めた濃度推移、上側が目標値から求めた理論的濃度推移。

Python プログラムを次ページ以降に示す。変数 SemiLog でグラフを選択できる。





以上

```
# Sawchuk-Zaske Method 2202.04.30
#
import matplotlib.pyplot as plt
import numpy as np

# Model Function for S.Zaske (1-comp. inf.)
def model_func(tt, Tinf, Rate, Tau, Vd, Ke):
    if tt <= Tinf:
        cp = Rate / Vd / Ke * (1 - np.exp(-Ke * tt))
        cp = cp / (1 - np.exp(-Ke * Tau))
    else:
        cp = Rate / Vd / Ke
        cp = cp * (1 - np.exp(-Ke * Tinf)) * np.exp(-Ke * (tt - Tinf))
        cp = cp / (1 - np.exp(-Ke * Tau))
    return cp
#
if __name__ == '__main__':
    # Set data
    zRate = 120
    zTinf = 1.0
    zTau  = 12.0
    CpTP = 40.0
    CpTT = 5.0

    x0_data = 0.0
    y0_data = 1.8
    Np = 2
    x_data = [3, 6]
    y_data = [10.5, 2.5]
    y_logdata = [np.log(10.5), np.log(2.5)]

    # Linear Regression
    a, b = np.polyfit(x_data, y_logdata, 1)
    zKe = -a
    zCmax = np.exp(a * zTinf + b)

    zVd1 = zRate * (1 - np.exp(-zKe * zTinf))
    zVd2 = (zCmax - y0_data * np.exp(-zKe * zTinf)) * zKe
```

```
zVd  = zVd1 / zVd2
zCL = zKe * zVd

rTau = -1 / zKe * np.log(CpTT / CpTP) + zTinf
rRate = zKe * zVd * CpTP
rRate = rRate * (1 - np.exp(-zKe * rTau)) / (1 - np.exp(-zKe * zTinf))

# Output
# Text output
print ("== Sawchuk-Zaske Method ==")
print ("-- Dosing Condition --")
print (' Inf. Rate = {:.1f}'.format(zRate))
print (' Inf. Time = {:.1f}'.format(zTinf))
print (' Interval. = {:.1f}'.format(zTau))
print ("-- Obserbed Data --")
print (' Trough = {:.2f}'.format(y0_data))
for i in range(Np):
    print (' {:}-th data'.format(i + 1))
    print ('  Time  = {:.1f}'.format(x_data[i]))
    print ('  Conc. = {:.2f}'.format(y_data[i]))
print ('-- Estimated Parameters --')
print ('  Ke = {:.3f}'.format(zKe))
print ('  Vd = {:.3f}'.format(zVd))
zCL = zVd * zKe
print ('  CL = {:.3f}'.format(zCL))

rTau = -1 / zKe * np.log(CpTT / CpTP) + zTinf
rRate = zKe * zVd * CpTP
rRate = rRate * (1 - np.exp(-zKe * rTau))
rRate = rRate / (1 - np.exp(-zKe * zTinf))

print ("-- Recommendation --")
print ('  Interval  = {:.2f}'.format(rTau))
print ('  Inf. Rate = {:.2f}'.format(rRate))
print ('  * target peak   = {:.1f}'.format(CpTP))
print ('  * target trough = {:.1f}'.format(CpTT))

# Plots
```

```
Nsim = 1024
x_min = 0.0
x_max = 12.0
# Semi-log plot (0: no, 1:yes)
SemiLog = 1
ax = plt.gca()
if SemiLog == 1:
    y_min = 0.1
    plt.ylabel("Log(Conc.)", fontsize = 12)
    ax.set_yscale('log')
else:
    y_min = 0.0
    plt.ylabel("Conc.", fontsize = 12)
    ax.set_yscale('linear')
y_max = 50.0

xs = [0] * (Nsim + 1)
ys = [0] * (Nsim + 1)
plt.title("Sawchuk-Zaske Method", fontsize = 12)
plt.xlim(x_min - 0.5, x_max); plt.ylim(y_min, y_max)
plt.xlabel("Time ", fontsize = 12)
# y-axis
plt.grid(True, which = "both", linestyle = "--", color = "gray")
# Target Range
plt.hlines([CpTP, CpTT], x_min, x_max,
          label = "Range", linestyle = 'solid', color ='blue')
# Observed data
plt.plot(x0_data, y0_data, label = "Obs(trough)",
        marker = 's', markersize = 8, color ="blue")
for j in range(Np):
    plt.plot(x_data[j],  y_data[j],
        marker = '.', linestyle ='', markersize = 16, color = "orange")
# Fitted Curve
#tt = 0
for i in range(Nsim):
    xs[i] = float(x_min + i * (x_max - x_min) / (Nsim - 1))
    tt = xs[i]
    ys[i] = model_func(tt, zTinf, zRate, zTau, zVd, zKe)
```

```
plt.plot(xs, ys, label = "Pred.", marker = 'None',
        linestyle = 'solid', color = "green")
# Recommended Curve
#tt = 0
for i in range(Nsim):
    xs[i] = float(x_min + i * (x_max - x_min) / (Nsim - 1))
    tt = xs[i]
    ys[i] = model_func(tt, zTinf, rRate, rTau, zVd, zKe)
plt.plot(xs, ys, label = "Recommended", marker = 'None',
        linestyle = 'solid', color ="red")
plt.legend(loc="upper right")
plt.show()
```