冪乗回帰 $y = a * x^b$

青木繁伸

2020 年 3 月 17 日

1　目的

冪乗回帰曲線 $y = a * x^b$ への当てはめを行う。

両辺の対数をると $\log y = \log a + b \log x$ のようになるので，従属変数 y の対数を取ったもの $\log y$ を独立変数の対数を取ったもの $logx$ で予測する単回帰式を解けばよい。

2　使用法

```
import sys
sys.path.append("statlib")
from multi import mreg4
```

2.1　引数

x	独立変数ベクトル（リストでもよい）
y	従属変数ベクトル（リストでもよい）
verbose	必要最小限のプリント出力をする（デフォルトは True）。

2.2　戻り値の名前

"b"	a
"b"	b
"predicted"	予測値
"resid"	残差（観察値 – 予測値）
"x"	独立変数 x
"y"	従属変数 y
"method"	分析手法名

3　使用例

```
import sys
sys.path.append("statlib")
from multi import power

import matplotlib.pyplot as plt

def graph(x, y, a, b):
    x0 = np.min(x)
    x1 = np.max(x)
    delta = (x1-x0)*0.05
    x2 = np.arange(max(x0-delta, 0), x1+delta, (x1-x0+2*delta)/500)
    y2 = a * x2**b
    plt.scatter(x, y, c="black", s=9)
    plt.plot(x2, y2, linewidth=0.5, color="red")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.show()
```

3.1 使用例 1

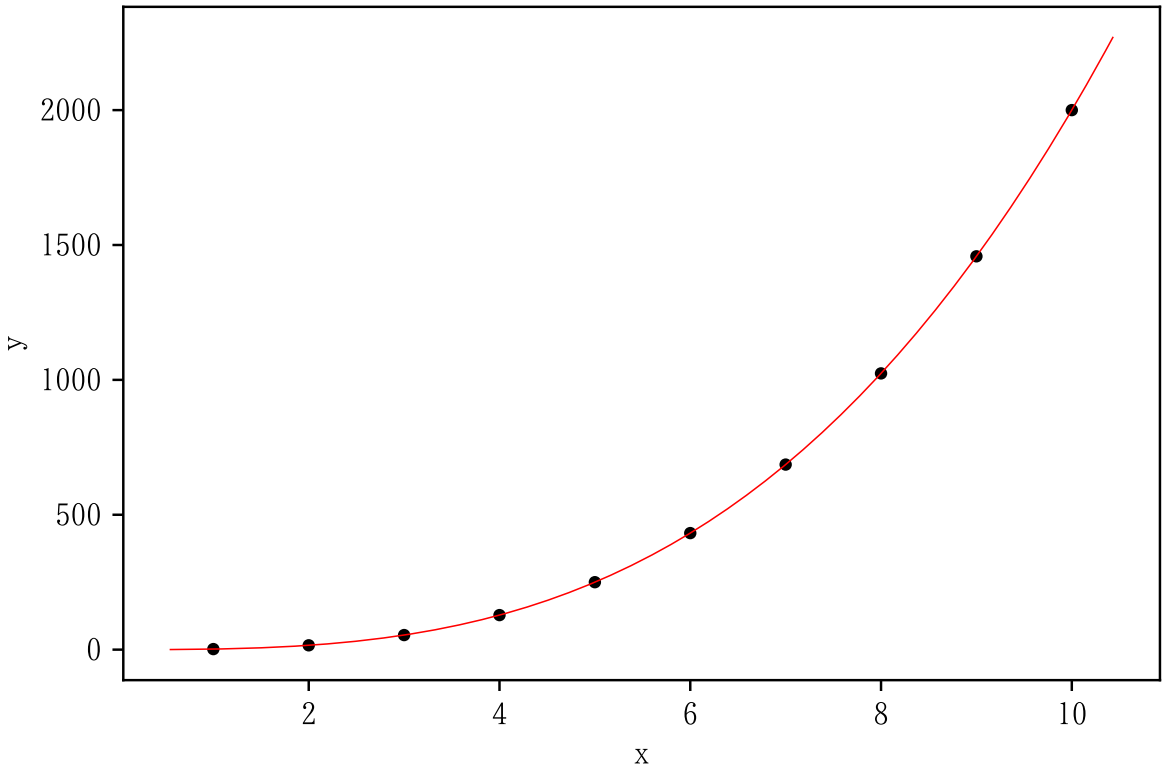
```
import numpy as np

x = np.arange(1, 11)
y = np.array([2, 16, 54, 128, 250, 432, 686, 1024, 1458, 2000])
ans = power(x, y)
```

```
y = a * x**b
a = 2, b = 3
```

	x	y	pred.	resid.
0	1	2	2.0	-3.552714e-15
1	2	16	16.0	-2.131628e-14
2	3	54	54.0	-6.394885e-14
3	4	128	128.0	-1.421085e-13
4	5	250	250.0	-2.842171e-13
5	6	432	432.0	-3.979039e-13
6	7	686	686.0	-6.821210e-13
7	8	1024	1024.0	-9.094947e-13
8	9	1458	1458.0	-1.136868e-12
9	10	2000	2000.0	-1.591616e-12

```
graph(x, y, ans["a"], ans["b"])
```



3.2 使用例 2

データによっては，当てはめが十分ではない場合がある。

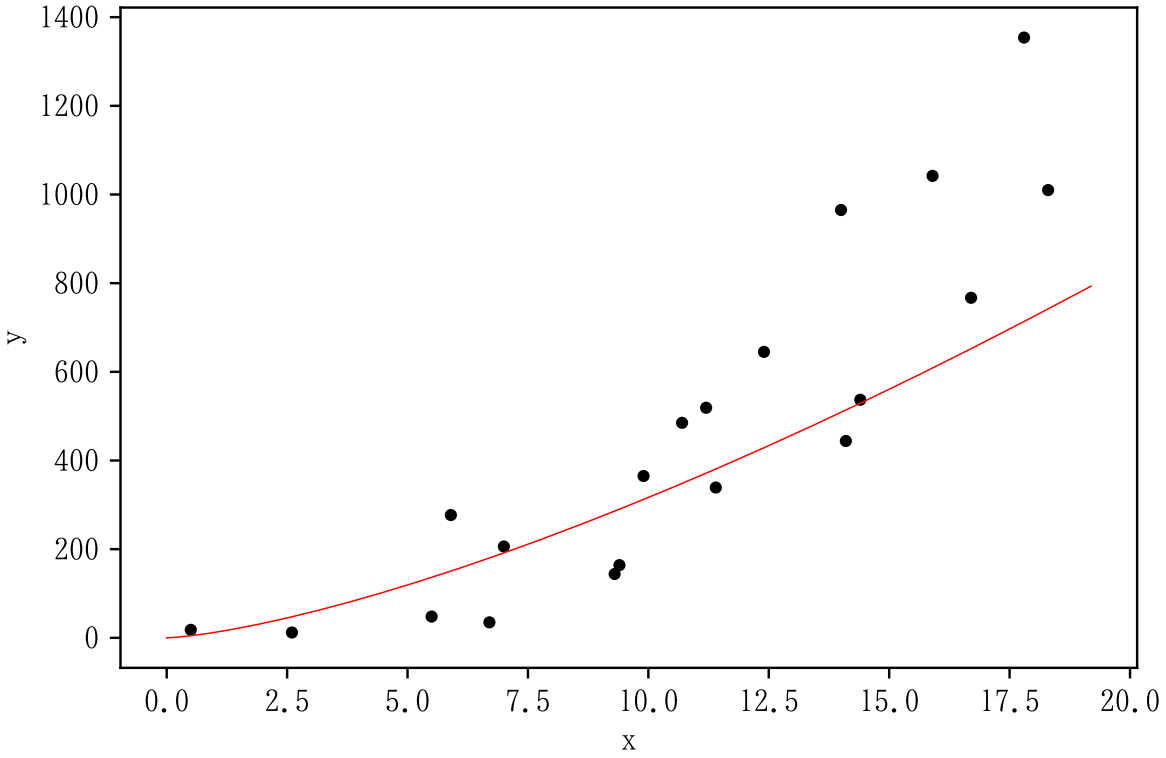
```
x2 = [0.5, 2.6, 5.5, 7, 9.4, 9.9, 11.4, 12.4, 14.1, 14, 15.9, 16.7,
17.8, 18.3, 14.4, 11.2, 5.9, 6.7, 9.3, 10.7]
y2 = [18, 12, 48, 206, 164, 365, 339, 645, 444, 965, 1042, 767, 1354,
1010, 537, 519, 277, 35, 144, 485]

ans2 = power(x2, y2)
graph(x2, y2, ans2["a"], ans2["b"])
```

```
y = a * x**b
a = 12.3348, b = 1.40932
```

	x	y	pred.	resid.
0	0.5	18	4.643932	13.356068
1	2.6	12	47.419937	-35.419937
2	5.5	48	136.313050	-88.313050
3	7.0	206	191.488482	14.511518
4	9.4	164	290.119647	-126.119647
5	9.9	365	312.102394	52.897606
6	11.4	339	380.754827	-41.754827
7	12.4	645	428.656271	216.343729

```
8   14.1   444  513.742385  -69.742385
9   14.0   965  508.614917  456.385083
10  15.9  1042  608.528206  433.471794
11  16.7   767  652.118338  114.881662
12  17.8  1354  713.459700  640.540300
13  18.3  1010  741.865292  268.134708
14  14.4   537  529.213983    7.786017
15  11.2   519  371.374642  147.625358
16   5.9   277  150.489614  126.510386
17   6.7    35  180.025033 -145.025033
18   9.3   144  285.779453 -141.779453
19  10.7   485  348.224661  136.775339
```



非線形回帰を行うと，改善されることもある。

```
from multi import nonlinear_fitting
ans3 = nonlinear_fitting(x2, y2, [1, 1], model="Power1")
graph(x2, y2, ans3["p"][0], ans3["p"][1])
```

```
Power1      by Marquardt method

                           estimates
a                           2.769506
b                           2.079742
residual sum of squares  457037.881264
```

	x	y	pred.	resid.
0	0.5	18.0	0.655145	17.344855
1	2.6	12.0	20.204123	-8.204123
2	5.5	48.0	95.976742	-47.976742
3	7.0	206.0	158.485143	47.514857
4	9.4	164.0	292.588726	-128.588726
5	9.9	365.0	325.887017	39.112983
6	11.4	339.0	437.010792	-98.010792
7	12.4	645.0	520.520418	124.479582
8	14.1	444.0	679.957934	-235.957934
9	14.0	965.0	669.966985	295.033015
10	15.9	1042.0	872.969095	169.030905
11	16.7	767.0	966.802175	-199.802175
12	17.8	1354.0	1103.961244	250.038756
13	18.3	1010.0	1169.433121	-159.433121
14	14.4	537.0	710.391767	-173.391767
15	11.2	519.0	421.216658	97.783342
16	5.9	277.0	111.064673	165.935327
17	6.7	35.0	144.685537	-109.685537
18	9.3	144.0	286.152392	-142.152392
19	10.7	485.0	383.049957	101.950043