Основные методы поиска оценок. Задача 3

Ильичёв А.С., 693

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
```

1. Считаем данные.

```
df = pd.read_csv('Weibull.csv', names=['val'])
```

2. Найдем логарифмическую функцию правдоподобия.

В качестве параметра θ выступает γ.

$$F(x) = 1 - e^{-x^\gamma} I(x > 0)$$

$$p(x) = F'(x) = \gamma x^{\gamma - 1} e^{-x^\gamma} I(x > 0)$$

$$f_\theta(x_1, \ldots, x_n) = \gamma^n \left(\prod_{i=1}^n x_i \right)^{\gamma - 1} e^{-\sum_{i=1}^n x_i^\gamma}$$

$$L_\theta(x_1, \ldots, x_n) = n \ln \gamma + (\gamma - 1) \sum_{i=1}^n \ln x_i - \sum_{i=1}^n x_i^\gamma$$

В силу монотонности логарифма для получения оценки максимального правдоподобия достаточно найти $\underset{\theta}{\operatorname{argmax}} L_\theta(x_1, \ldots, x_n)$. Получим задачу

$$\underset{\theta}{\operatorname{argmax}} \left(n \ln \gamma + (\gamma - 1) \sum_{i=1}^n \ln x_i - \sum_{i=1}^n x_i^\gamma \right).$$

3. Обработаем выборку.

```
df.head()
```

	val
0	0.00

	val
1	0.01
2	0.11
3	1.79
4	0.03

```
len(df[df['val'] == 0])
```

```
400
```

Видим, что в выборке присутствуют нулевые значения, однако в функцию плотности могут входить отрицательные степени x. Поэтому заменим нули на значение, меньшее любого другого в выборке. Вообще говоря, в выборке больше 10% нулей, поэтому итоговая оценка будет зависеть от того, каким мы выберем это значение. Тем не менее, из-за округления всего лишь до второго порядка мы не знаем, какие именно элементы были в исходном распределении, поэтому замена тут достаточно произвольна.

```
df['val'] = df['val'].replace(0.0, df[df['val'] > 0]['val'].min() / 100)
```

```
df.head()
```

	val
0	0.0001
1	0.0100
2	0.1100
3	1.7900
4	0.0300

```
df.describe()
```

	val
count	3652.000000
mean	2.858970
std	7.819182
min	0.000100

	val
25%	0.050000
50%	0.465000
75%	2.262500
max	121.040000

4. Оценим параметр сдвига методом максимального правдоподобия.

```
def calc_func(df, g):
    # считаем максимизируемую функцию
    return (len(df) * g + (g - 1) * np.sum(np.log(df['val'])) -
            np.sum(df['val'] ** g))
```

Оценку производим по сетке в логарифмической шкале с шагом 10^{-3}.

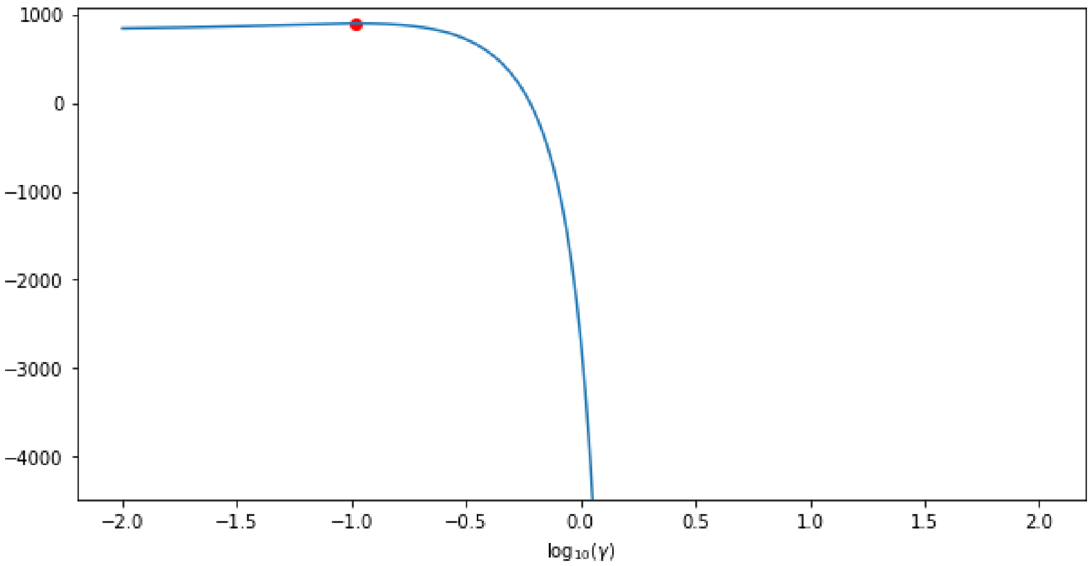
```
grid = np.linspace(-2, 2, 4 * 1000 + 1)
```

```
def find_max(df):
    vals = np.array([calc_func(df, 10 ** lg) for lg in grid])
    plt.figure(figsize=(10,5))
    plt.plot(grid, vals)
    max_i = np.argmax(vals)
    plt.scatter([grid[max_i]], [vals[max_i]], c='red')
    plt.xlabel(r'$\log_{10}(\gamma)$')
    plt.ylim(-vals[max_i] * 5, vals[max_i] * 1.2)
    plt.show()
    print('max = ', vals[max_i])
    print('log10(gamma) = ', grid[max_i])
    print('Оценка максимального правдоподобия: gamma = ', 10 ** grid[max_i])
```

а) По первым четырем годам (т.к. всего 3652 элемента, среди первых 4 годов есть один високосный):

```
df4 = df.head(365 * 4 + 1)
```

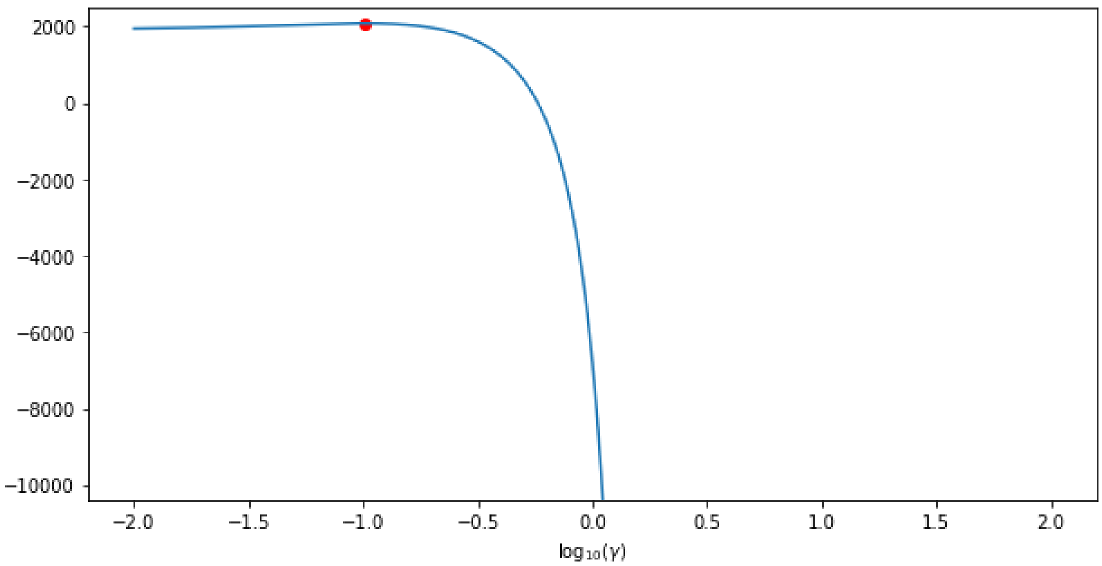
```
find_max(df4)
```




```
max =  896.2481143914201
log10(gamma) =  -0.98
Оценка максимального правдоподобия: gamma =  0.10471285480508996
```

б) По всей выборке:

```
find_max(df)
```




```
max =  2076.3371414938456
log10(gamma) =  -0.996
Оценка максимального правдоподобия: gamma =  0.10092528860766845
```

Вывод.

Полученные значения оценок отличаются, но не сильно. Так что в данном случае, если высокая точность оценки не требуется, можно рассматривать часть выборки, ускоряя вычисления.

В этой выборке большой максимальный элемент (121.04), да и 0.75-квантиль больше единицы. Поэтому при приближении $\log_{10}(\gamma)$ к нулю (а γ к 1) и дальнейшем увеличении максимизируемая функция резко падает (ее последнее слагаемое, перед которым стоит минус, растет показательно).