# Fit data with a lognormal function via Maximum Likelihood estimators

Asked 4 days ago Modified today Viewed 104 times

0

**The bounty expires in 5 days**. Answers to this question are eligible for a +100 reputation bounty. d.b wants to **draw more attention** to this question.

Could someone help me in fitting the data `collapse_fractions` with a lognormal function, which has median and standard deviation derived via the maximum likelihood method?

I tried `scipy.stats.lognormal.fit(data)`, but I did not obtain the data I retrieved with Excel. The excel file can be downloaded: `https://stacks.stanford.edu/file/druid:sw589ts9300/p_collapse_from_msa.xlsx`

Also, any reference is really welcomed.

```
import numpy as np

intensity_measure_vector = np.array([[0.2, 0.3, 0.4, 0.6, 0.7, 0.8, 0.9, 1]])
no_analyses = 40
no_collapses = np.array([[0, 0, 0, 4, 6, 13, 12, 16]])
collapse_fractions = np.array(no_collapses/no_analyses)

print(collapse_fractions)
# array([[0.   , 0.   , 0.   , 0.1  , 0.15 , 0.325, 0.3  , 0.4  ]])

collapse_fractions.shape
# (1, 8)

import matplotlib.pyplot as plt
plt.scatter(intensity_measure_vector, collapse_fractions)
```

python numpy scipy.stats

Share Edit Delete Flag

edited Jul 22 at 17:27
d.b
**31.5k** 5 32 72

asked Jul 21 at 18:27
Giuseppe Degan Di Dieco
**1** 2

New contributor

1 What did you obtain with excel? – d.b Jul 21 at 18:38

Good evening d.b, I obtained a median of 1.08, and a deviation of 0.43. I think it'd good to post the Excel file as part of my question. I'm new to python, so not very practical with it. With .fit(data), I obtained this: fitting_parameters = lognorm.fit(collapse_fractions.T) print(fitting_parameters) (7.922896498137963, -1.9180772982941385e-20, 0.17060313722466533) which should be shape, loc, scale. I think it's my fault however, since I

don't have a proper theoretical understanding of involved equations.
– Giuseppe Degan Di Dieco Jul 21 at 18:56

You have zero in your data which causes problems when fitting `lognormal`. As for the explanation of parameters, see stackoverflow.com/q/8870982/7128934 – d.b Jul 22 at 18:59

Good morning d.b., many thanks for your help. I had a quick look at your suggested link, and it seems great content. I'll now try to understand it, and keep you posted. Many thanks.
– Giuseppe Degan Di Dieco Jul 23 at 8:54

## 2 Answers

Sorted by:
Trending sort available
Highest score (default)

1

I couldn't figure out how to get the `lognorm.fit` to work. So I just implemented the functions from your excel-file and used `scipy.optimize` as the optimizer. The added benefit is, that it is easier to understand what is actually going on compared to `lognorm.fit` especially with the excel on the side.

Here is my implementation:

```
from functools import partial
import numpy as np
from scipy import optimize, stats


im = np.array([0.2, 0.3, 0.4, 0.6, 0.7, 0.8, 0.9, 1])
im_log = np.log(im)
number_of_analyses = np.array([40, 40, 40, 40, 40, 40, 40, 40])
number_of_collapses = np.array([0, 0, 0, 4, 6, 13, 12, 16])

FORMAT_STRING = "{:<20}{:<20}{:<20}"
print(FORMAT_STRING.format("sigma", "beta", "log_likelihood_sum"))


def neg_log_likelihood_sum(params, im_l, no_a, no_c):
    sigma, beta = params
    theoretical_fragility_function = stats.norm(np.log(sigma), beta).cdf(im_l)
    likelihood = stats.binom.pmf(no_c, no_a, theoretical_fragility_function)
    log_likelihood = np.log(likelihood)
    log_likelihood_sum = np.sum(log_likelihood)

    print(FORMAT_STRING.format(sigma, beta, log_likelihood_sum))
    return -log_likelihood_sum


neg_log_likelihood_sum_partial = partial(neg_log_likelihood_sum, im_l=im_log,
no_a=number_of_analyses, no_c=number_of_collapses)


res = optimize.minimize(neg_log_likelihood_sum_partial, (1, 1), method="Nelder-
Mead")
print(res)
```

And the final result is:

```
 final_simplex: (array([[1.07613697, 0.42927824],
       [1.07621925, 0.42935678],
       [1.07622438, 0.42924577]]), array([10.7977048 , 10.79770573,
10.79770723]))
           fun: 10.797704803509903
       message: 'Optimization terminated successfully.'
          nfev: 68
           nit: 36
        status: 0
       success: True
             x: array([1.07613697, 0.42927824])
```

The interesting part for you is on line one, the same final result as in the excel-calculation (sigma=1.07613697 and beta=0.42927824).

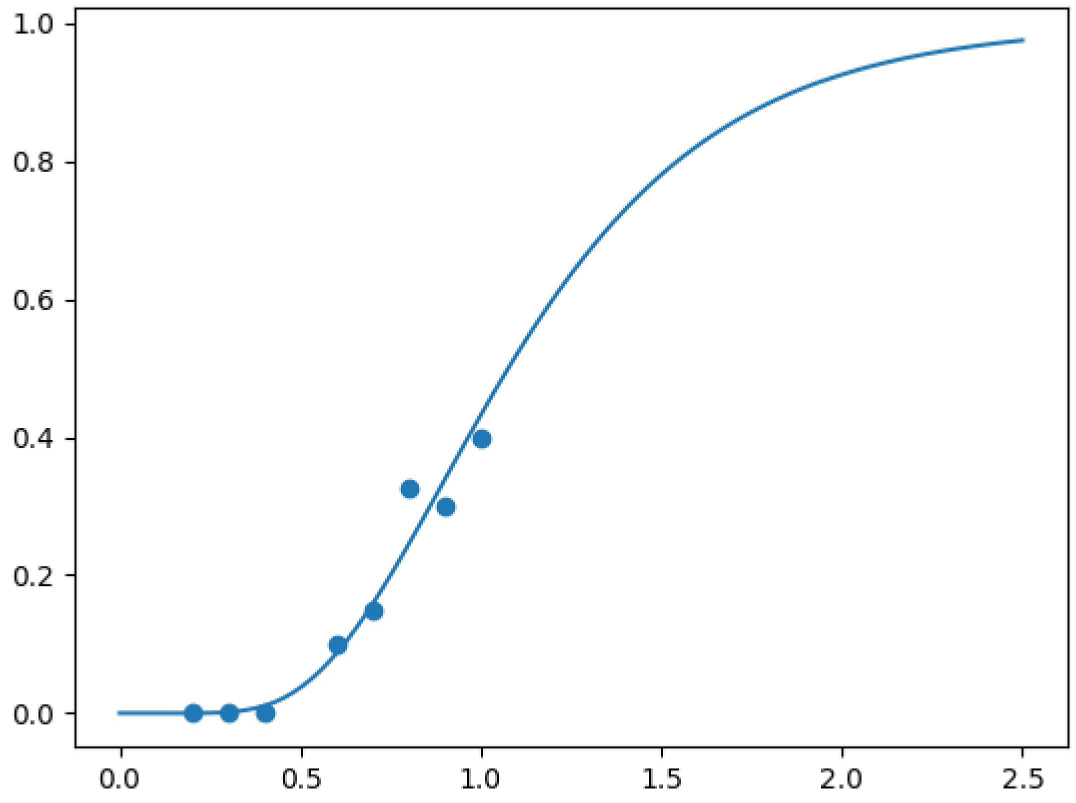
If you have any questions about what I did here, don't hesitate to ask as you said you are new to python. A few things in advance:

- I did minimize the negative likelihood-sum as there is no maximizer in scipy.
- partial from functools returns a function that has the specified arguments already defined (in this case `im_l`, `no_a` and `no_c` as they don't change) the partial function can then be called with just the missing argument.
- The `neg_log_likelihood_sum` -function is basically whats happening in the excel-file, so it should be easy to understand when viewing it side-by-side.
- `scipy.optimize.minimize` minimizes the function given as the first argument by changing the parameters (start-value as second argument). The method is chosen because it gave good results, I didn't dive deep into the abyss of different optimization-methods, gradients etc. So it may well be, that there is a better setup, but this one works fine and seems faster than the optimization with `lognorm.fit`.

The plot like in the excel-file is done like this with the results `res` from the optimization:

```
import matplotlib.pyplot as plt


x = np.linspace(0, 2.5, 100)
y = stats.norm(np.log(res["x"][0]), res["x"][1]).cdf(np.log(x))
plt.plot(x, y)
plt.scatter(im, number_of_collapses/number_of_analyses)
plt.show()
```



Share Edit Follow

edited 42 mins ago

answered 17 hours ago

MangoNrFive

**692** 2 15

Dear MangoNrFive, many thanks for your help. I saw your solution, and it's brilliant honestly. It deserves an oscar in my opinion. Unfortunately, yes, I'm new to Python, trained to use commercial software and double check results with simple Excel calculations. I tried the lognormal.fit as well, but, really, didn't figure out how it works. Then, I used the 'stats.lognorm(s, scale)', and after 'stas.lognorm.cdf(x)', but I manually changed s and scale. I'll study your solution, and get back to you with any doubts. In the meantime, many thanks, from a PhD student. --- Giuseppe
– Giuseppe Degan Di Dieco 4 mins ago Edit

**This post is hidden.** It was deleted yesterday by Ryan M ♦.

-1

Good evening all,

I tried to search online for available Python modules, but I couldn't find any, unfortunately.

I am trying to find mean and standard deviation of ln(variable) which is normal distributed, but my likelihood is given by a Binomial distribution.

Therefore, the ln(likelihood) should maximise the Binomial distribution, which has the parameter 'probability' function of mean and standard deviation of the ln(variable).

I did it successfully with Excel, but I couldn't find a way with Python.

Look forward to hearing from you.

Share Edit **Undelete** Flag

answered 2 days ago

Giuseppe Degan Di Dieco

**1** 2

New contributor

1 As it’s currently written, your answer is unclear. Please edit to add additional details that will help others understand how this addresses the question asked. You can find more information on how to write good answers in the help center. – Community Bot yesterday

2 Please use the edit link on your question to add additional information. The Post Answer button should be used only for complete answers to the question. - From Review – Blastfurnace yesterday

Comments disabled on deleted / locked posts / reviews