

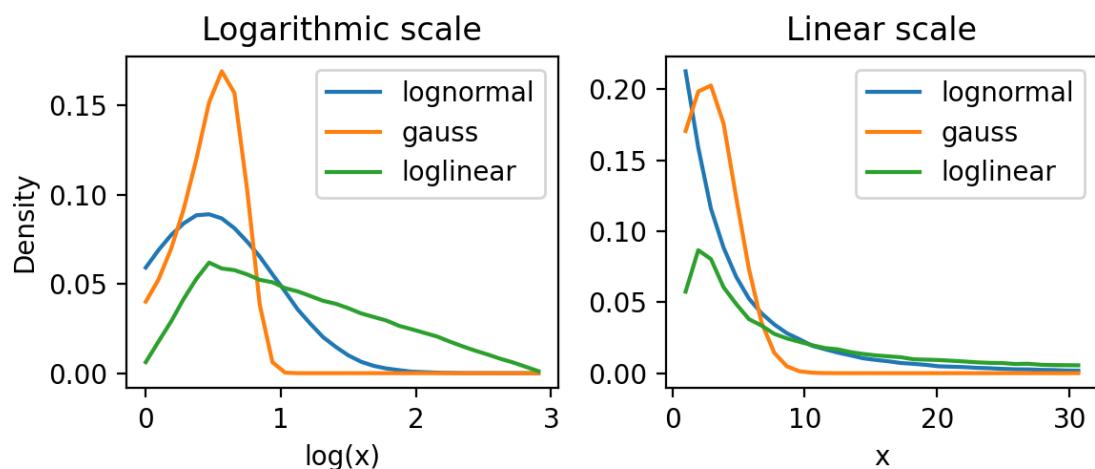
supermodel

January 12, 2022

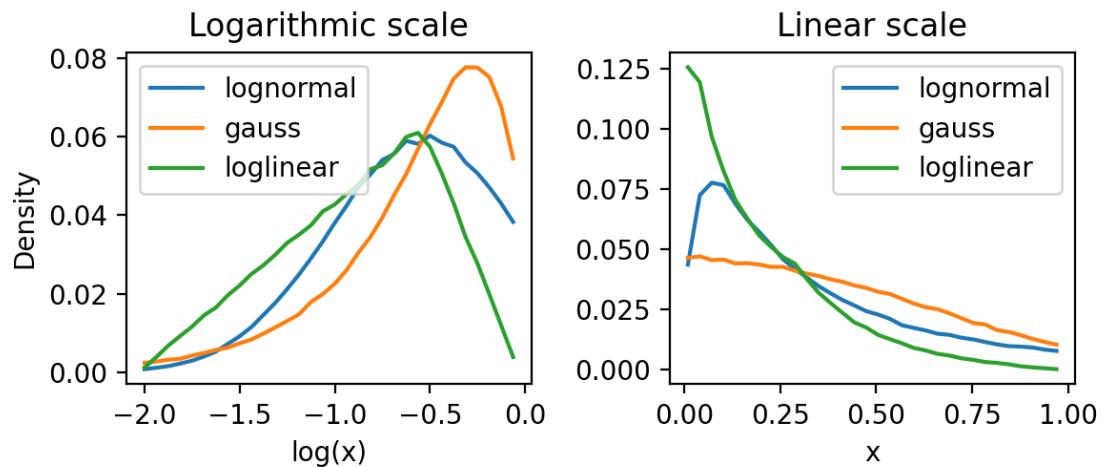
```
[1]: from pca import cluster, PCA
import numpy as np
import matplotlib.pyplot as plt
import warnings
import pandas as pd
from tabela_napake_p import modeli_napake
from mpl_toolkits.mplot3d import Axes3D
warnings.filterwarnings("ignore")

# primeri vseh treh porazdelitev v primeru povprečja večjega in manjšega od 1,
# ob tem so najprej predstavljene porazdelitve za N v primeru večje možnosti
# za širitev na druge planete
```

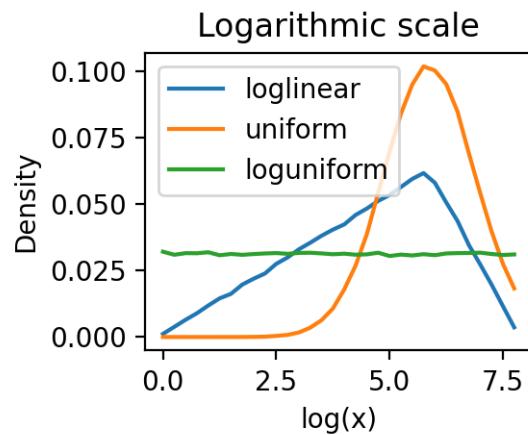
Distributions of x from 1 to 1000 with mean (peak) at 3.16



Distributions of x from 0.01 to 1 with mean (peak) at 0.32



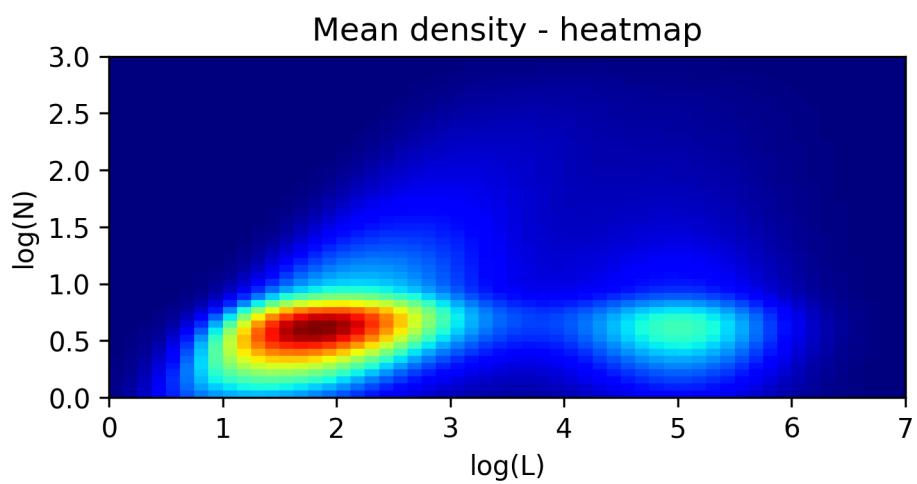
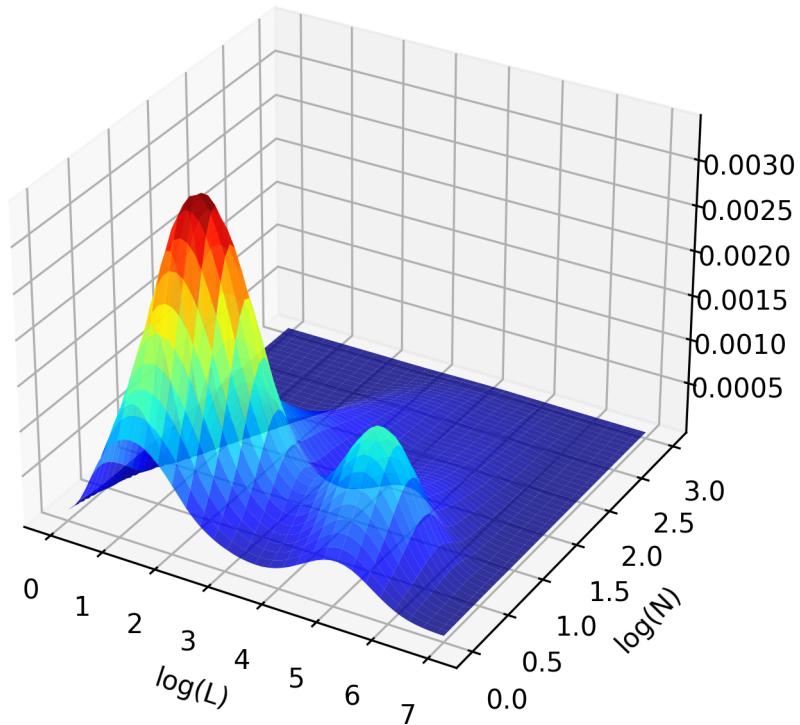
Distributions of x from 1 to $1e8$
with mean (peak) at $1e6$

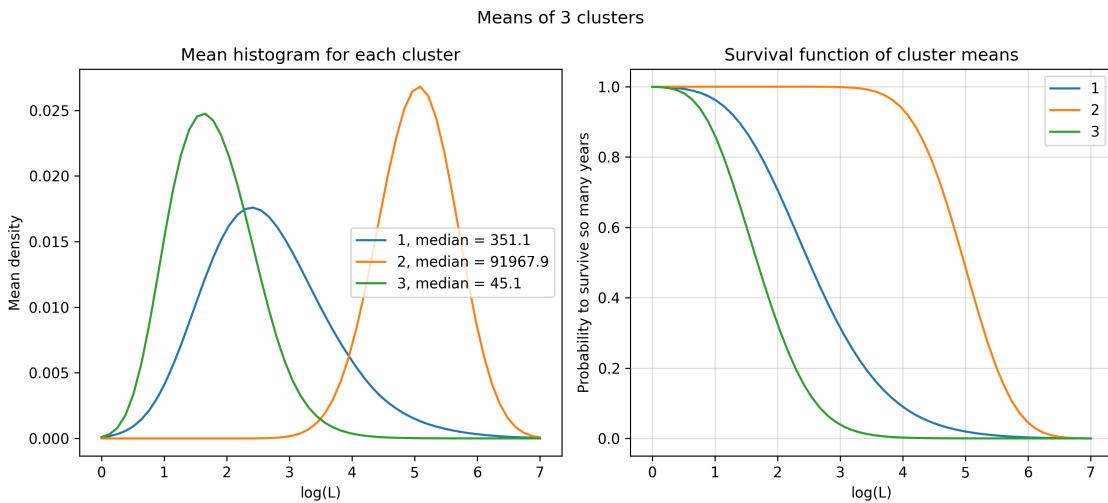
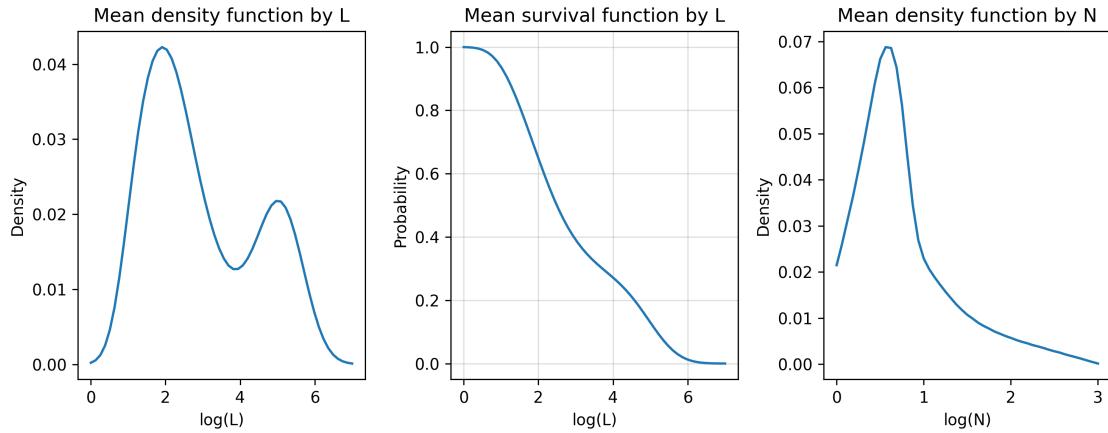


```
[2]: cluster(model=0, ks=[3, 5], supermodel=1) # supermodel  
cluster(model=0, ks=[5], supermodel=2) # supermodel 2
```

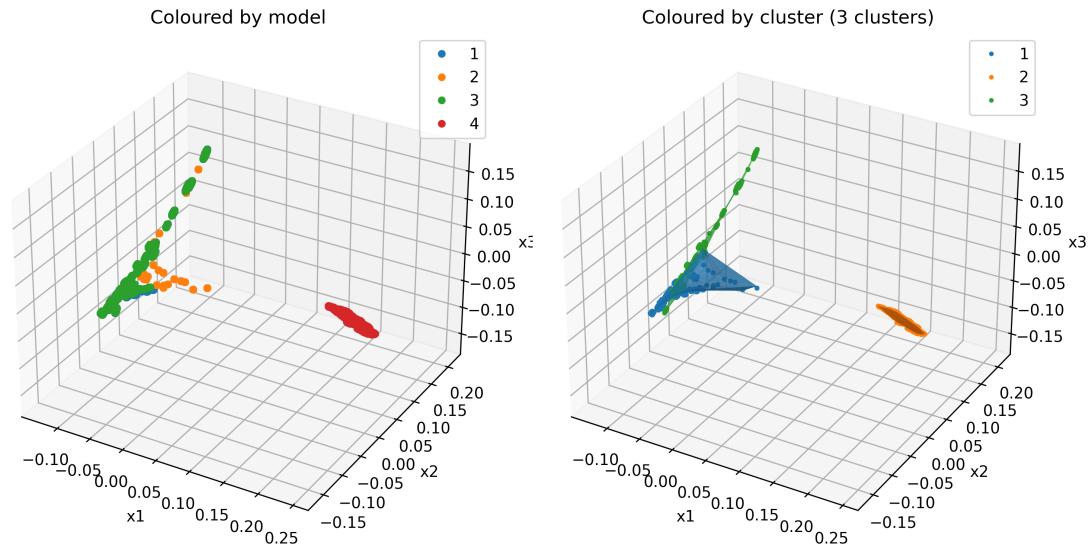
Supermodel

Mean density

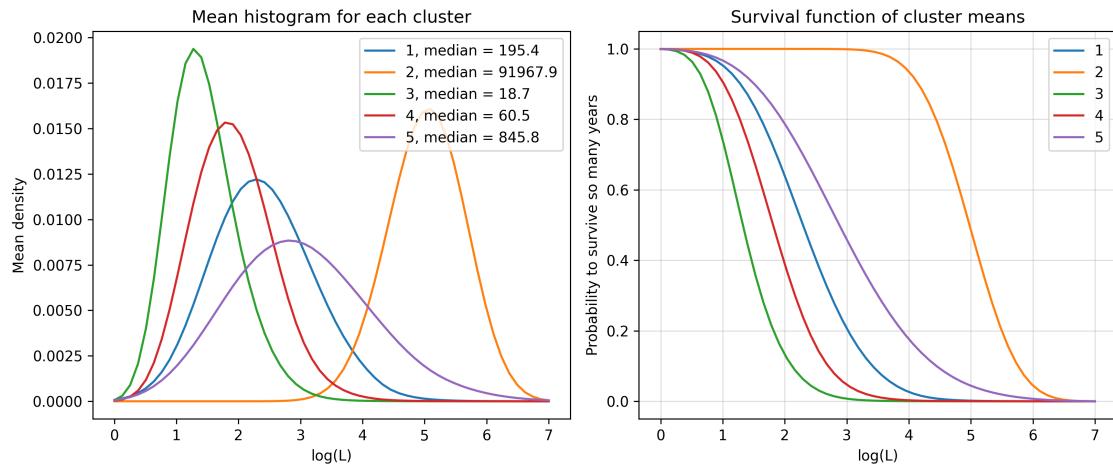




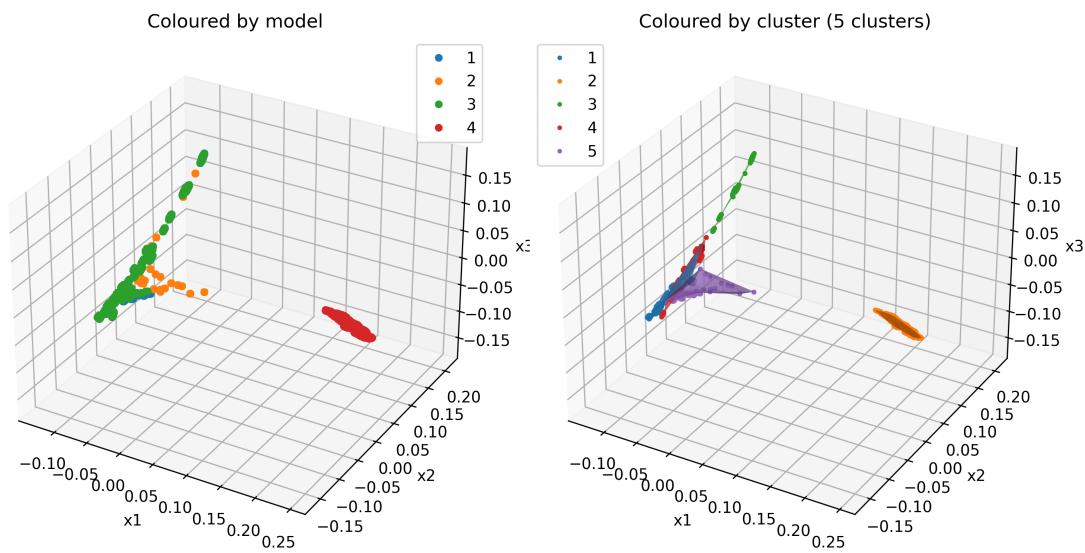
Supermodel after Principal component analysis (PCA)



Means of 5 clusters

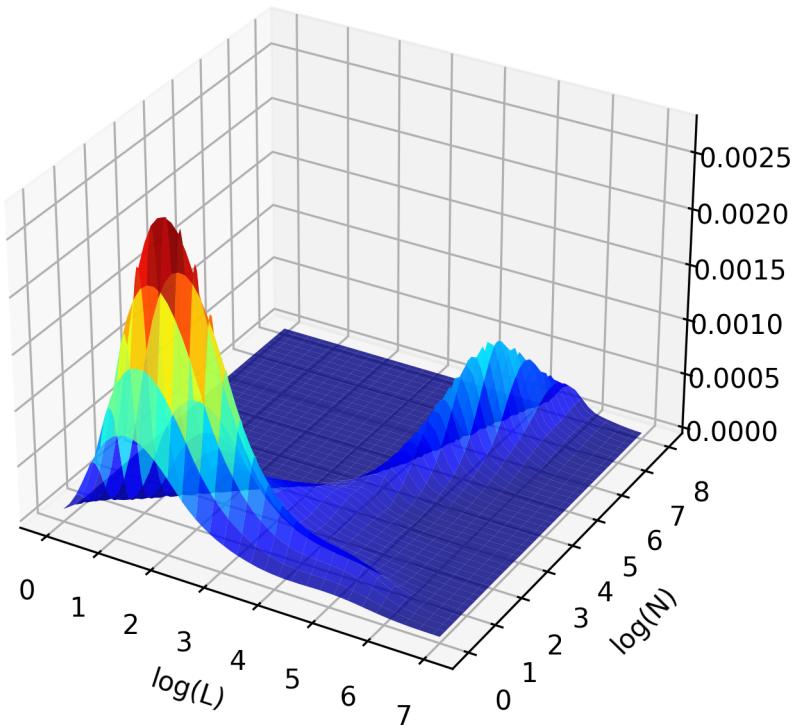


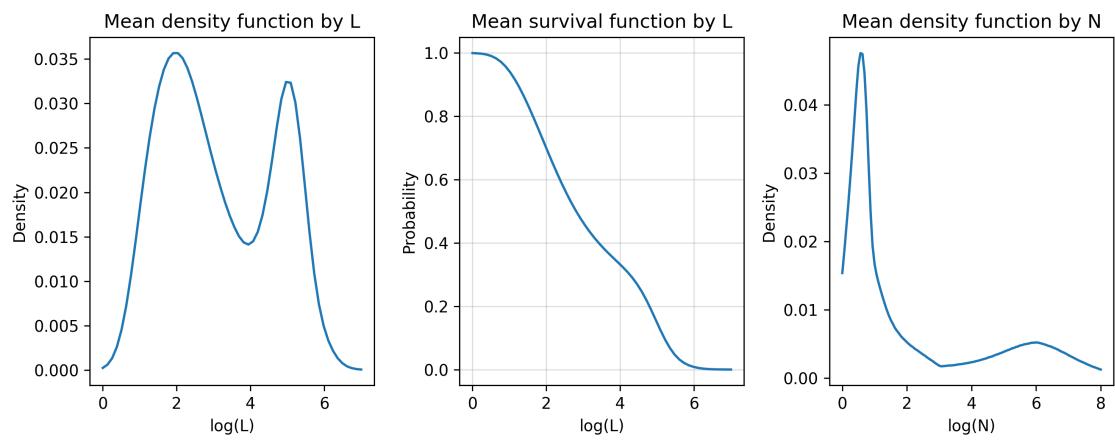
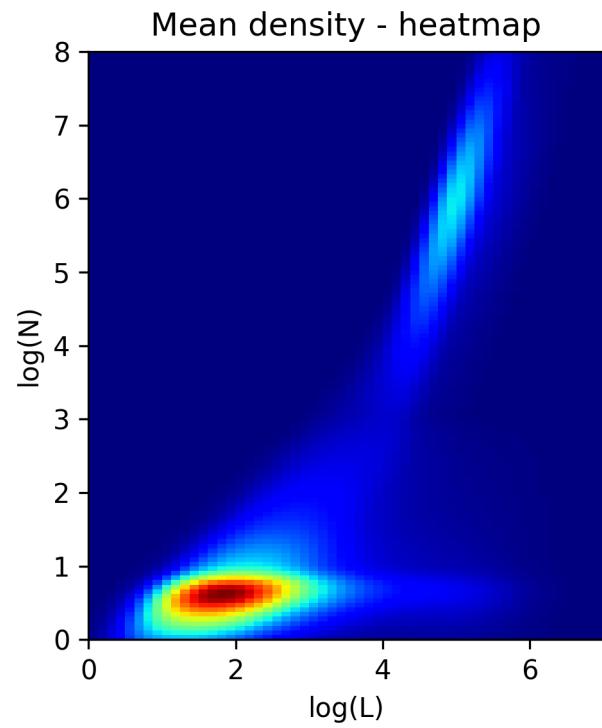
Supermodel after Principal component analysis (PCA)

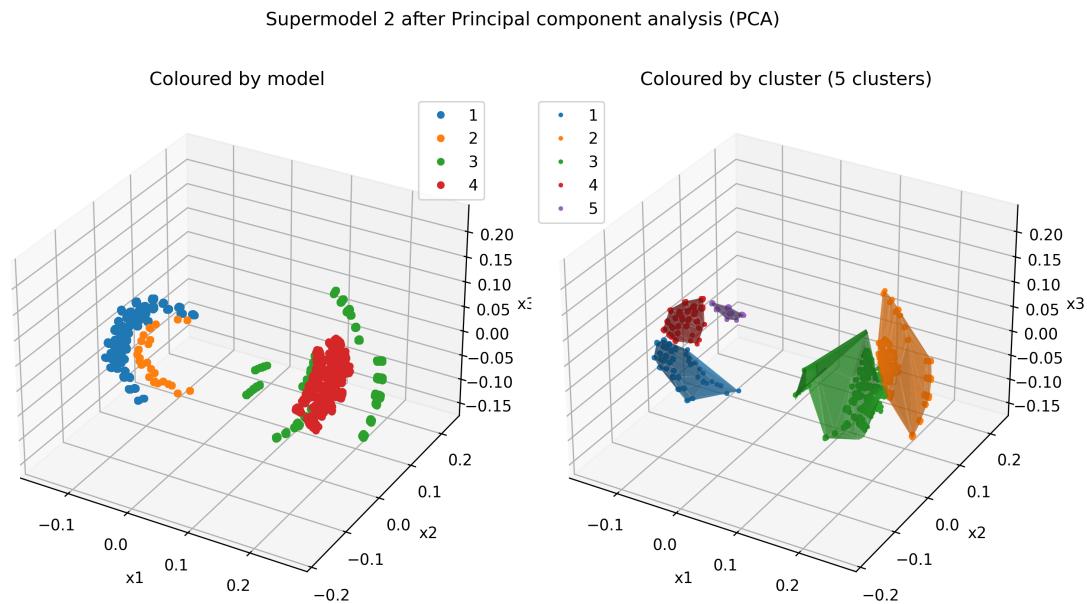
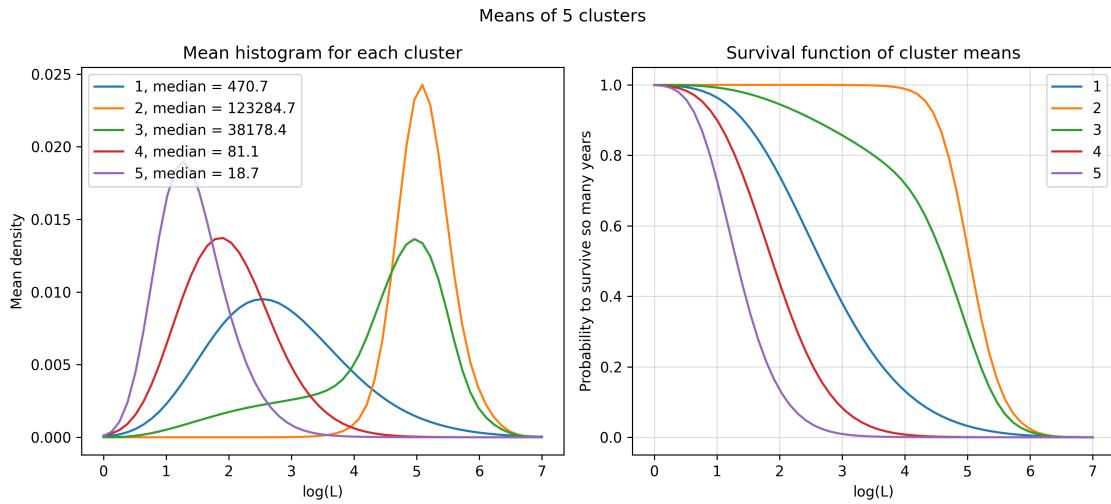


Supermodel 2

Mean density



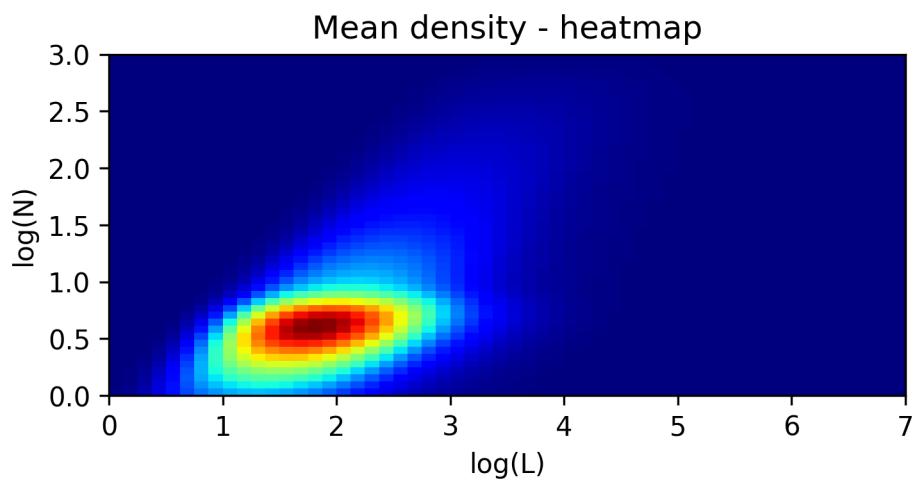
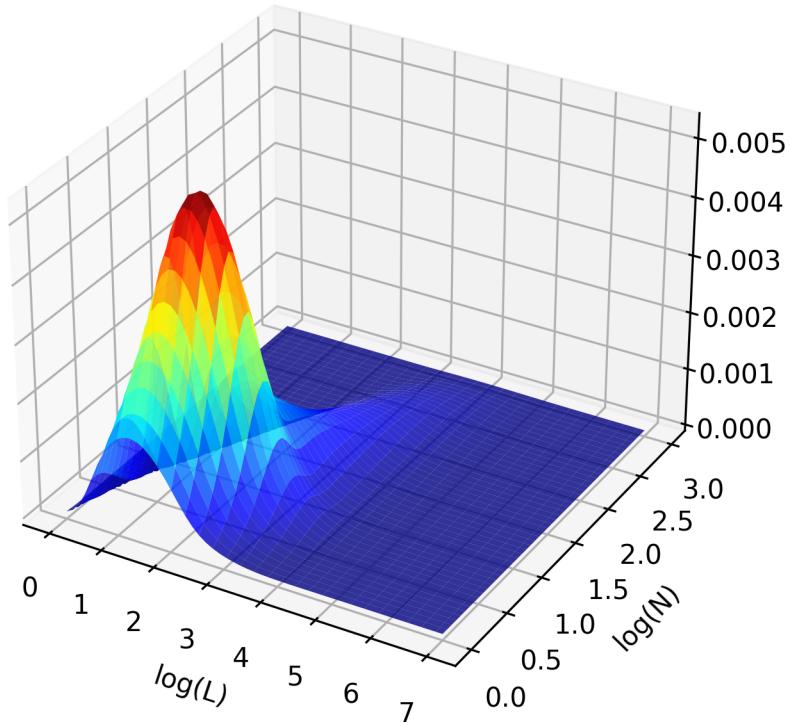


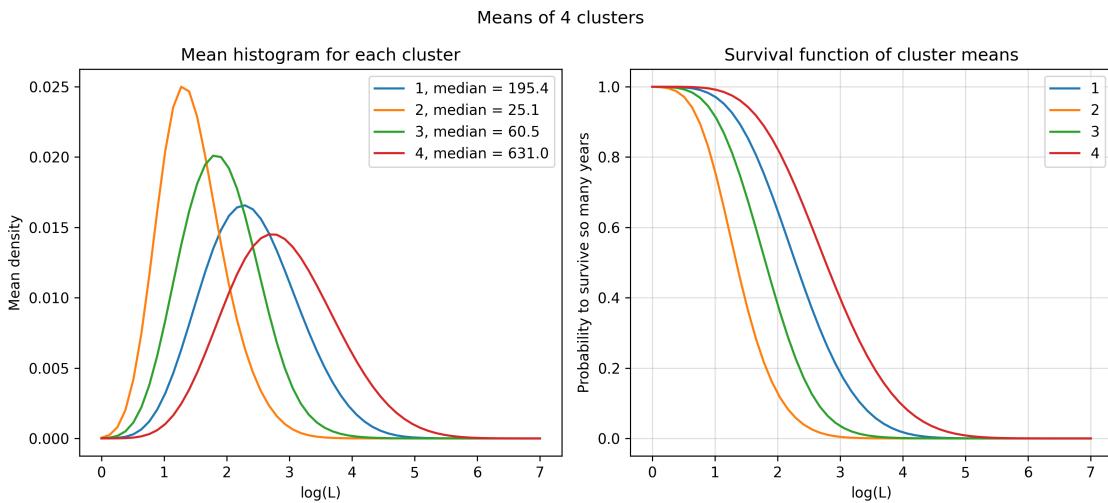
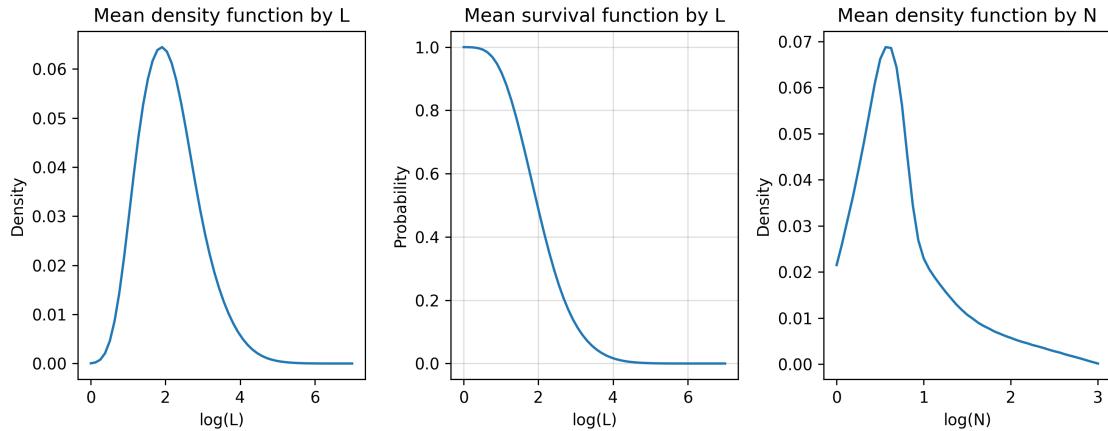


```
[3]: cluster(model=1, ks=[4]) # model 1
```

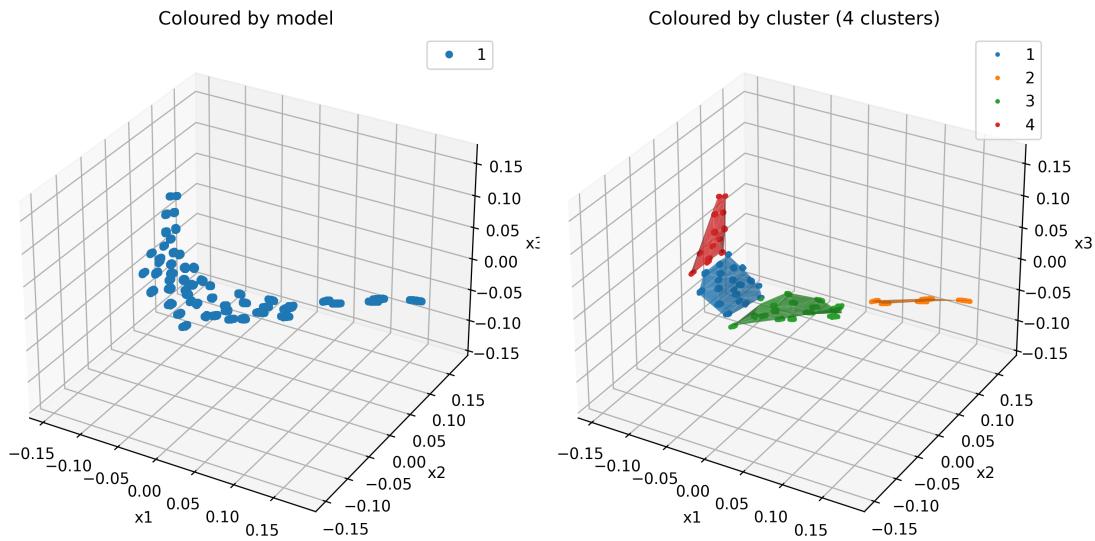
Model I

Mean density

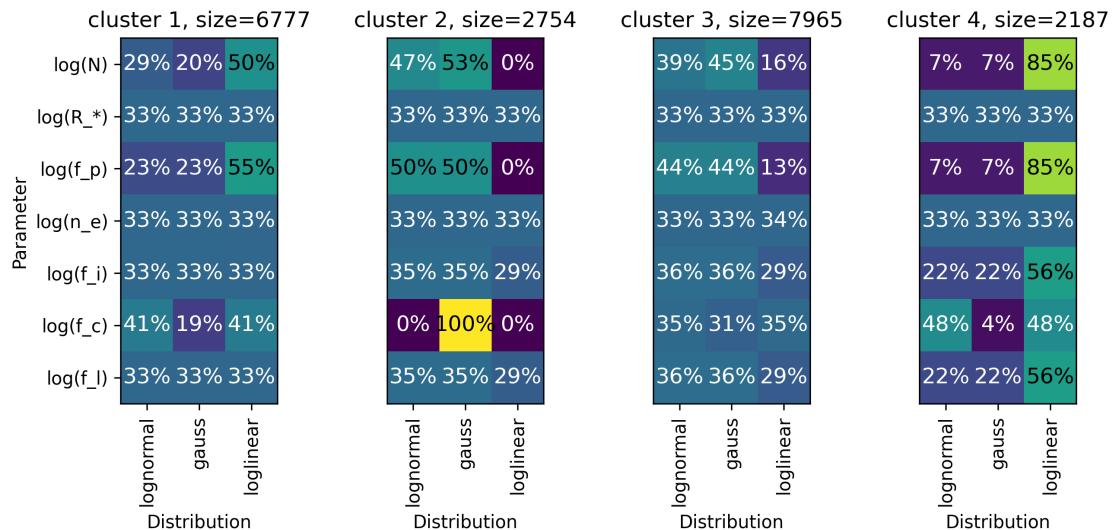




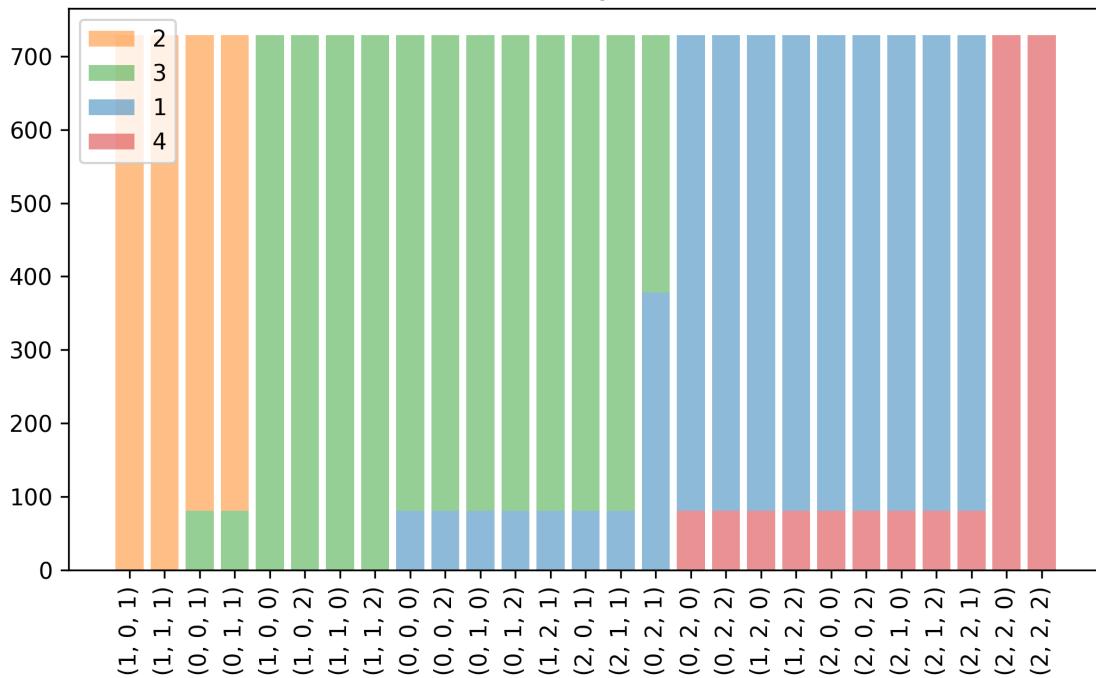
Model I after Principal component analysis (PCA)



Percent of submodels in model 1 distributed with particular distribution on particular parameter in each of 4 clusters



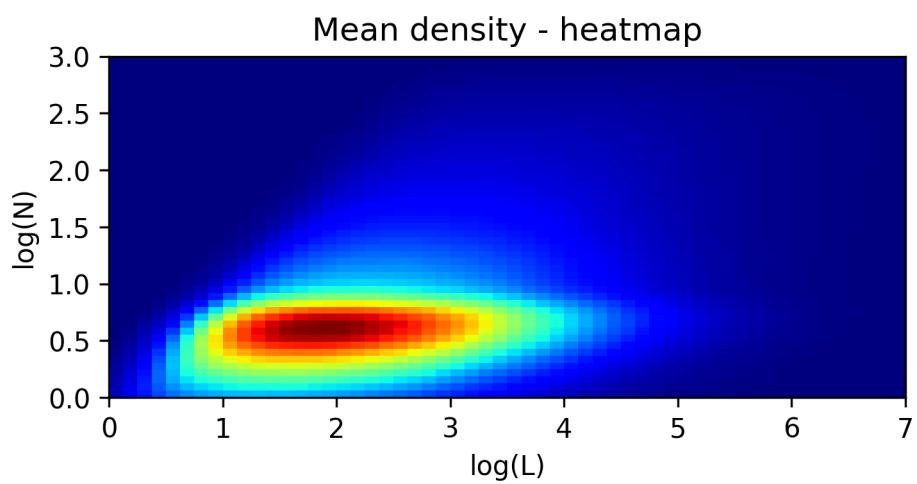
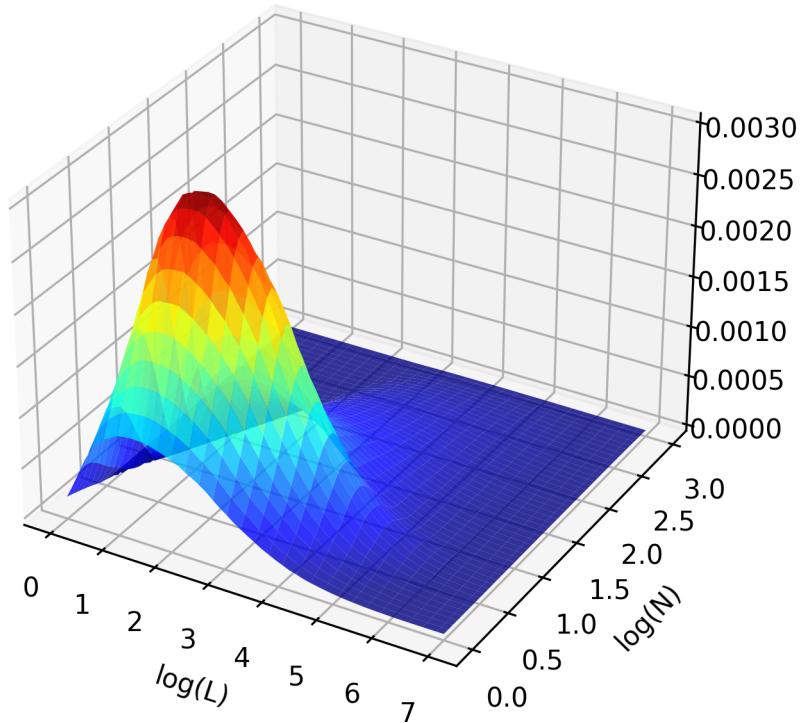
Percentage of distributions
 $(\log(N), \log(f_p), \log(f_c))$ by (lognormal, gauss, loglinear)
 coloured by cluster

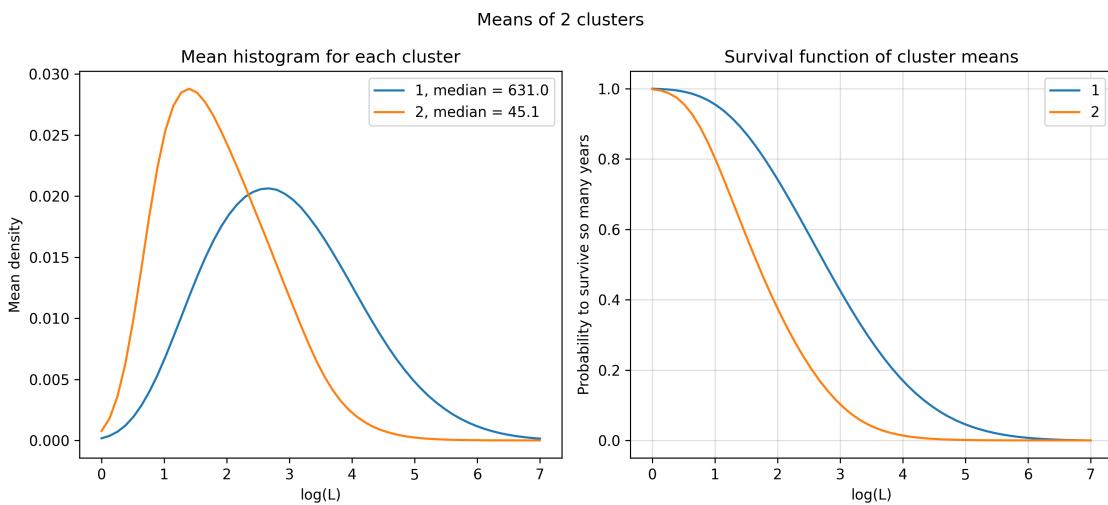
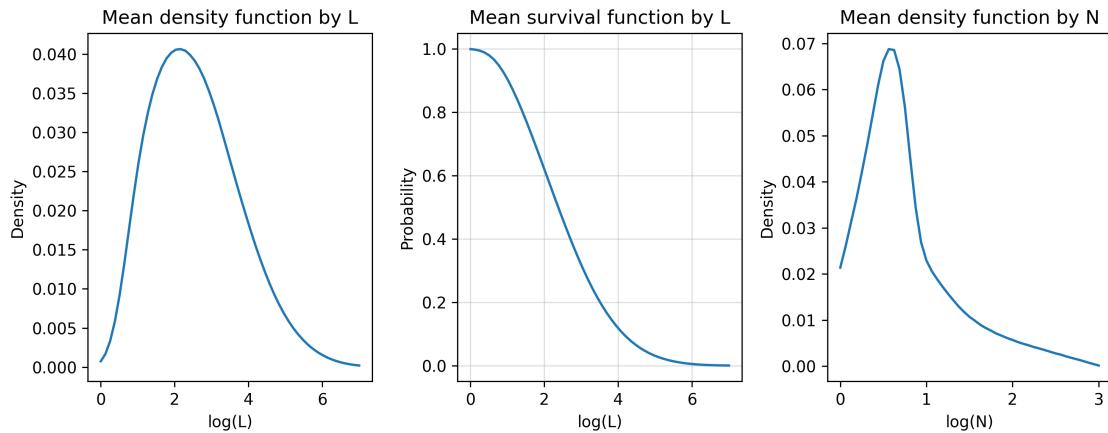


```
[4]: cluster(model=2, ks=[2]) # model 2
```

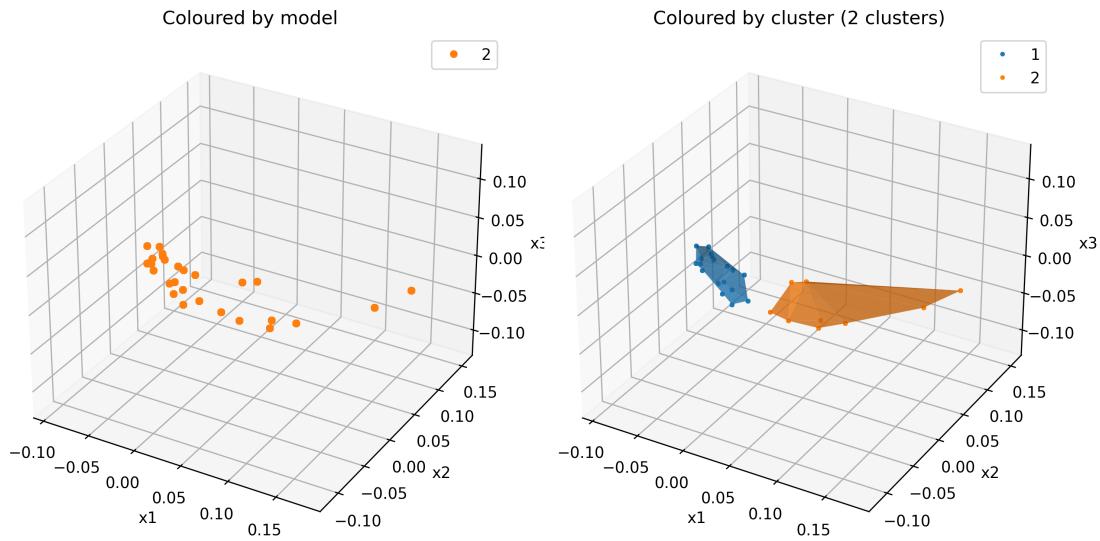
Model II

Mean density





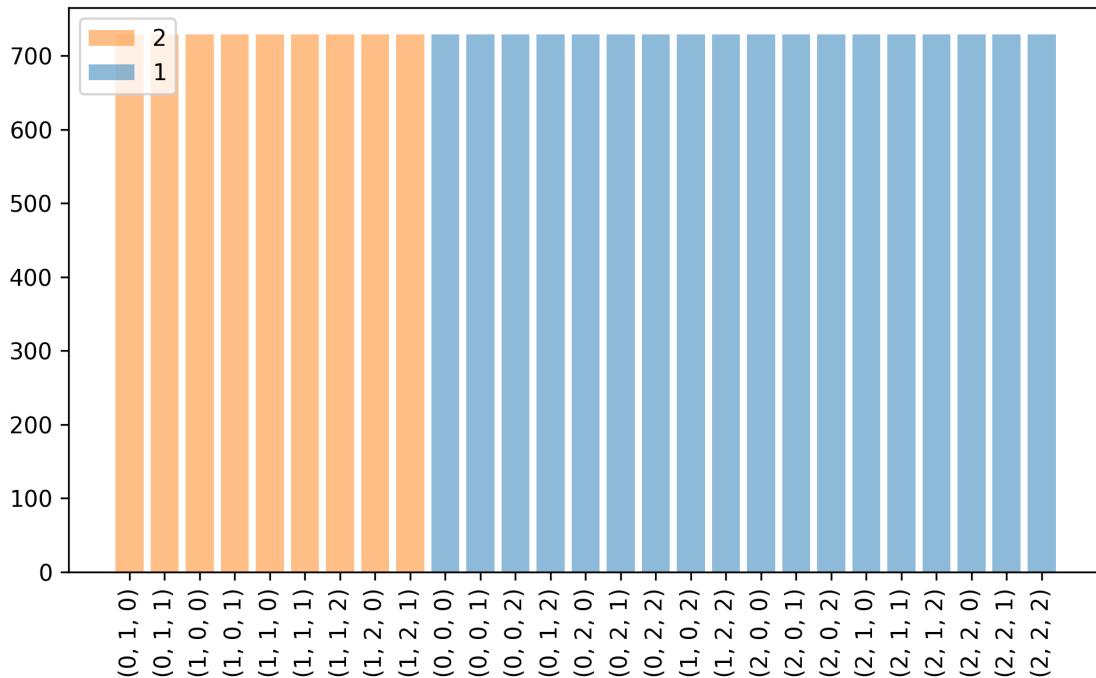
Model II after Principal component analysis (PCA)



Percent of submodels in model 2 distributed with particular distribution on particular parameter in each of 2 clusters



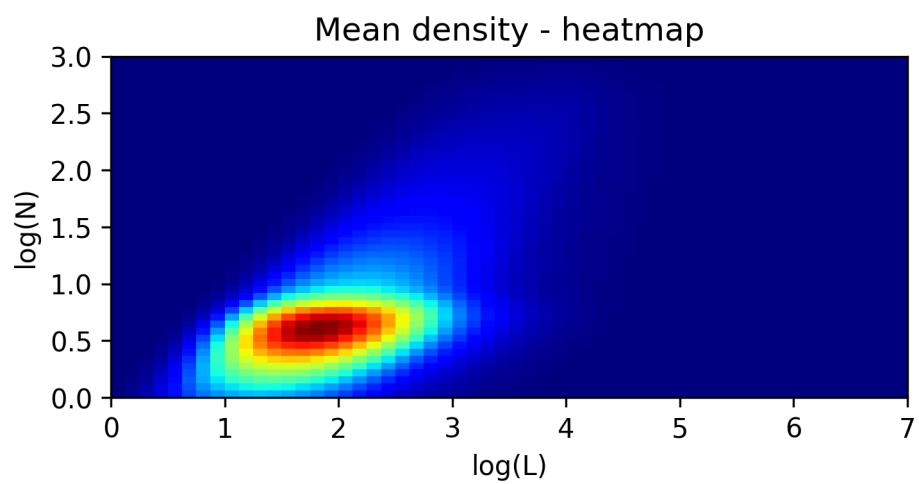
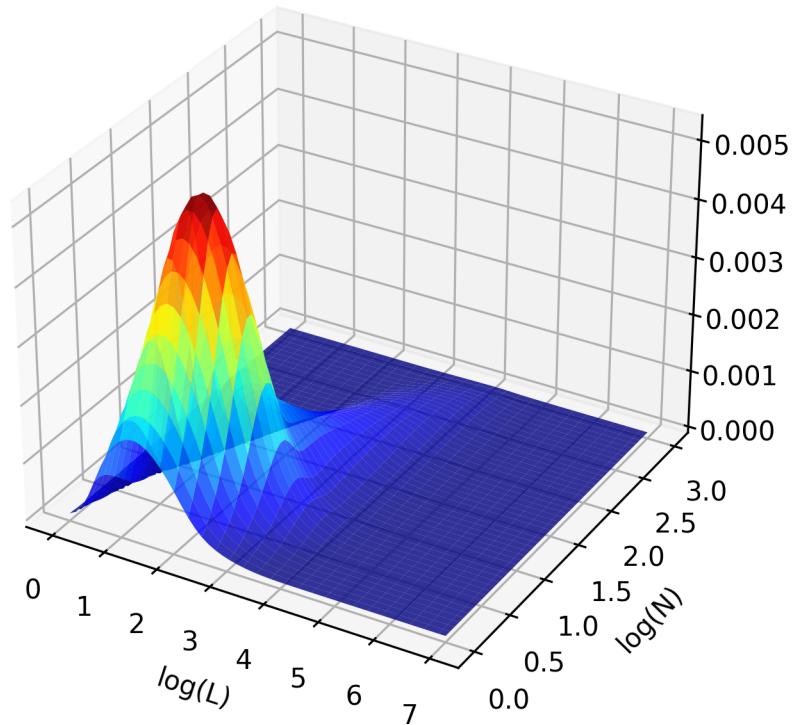
Percentage of distributions
 $(\log(f_b), \log(f_a), \log(N))$ by (lognormal, gauss, loglinear)
 coloured by cluster

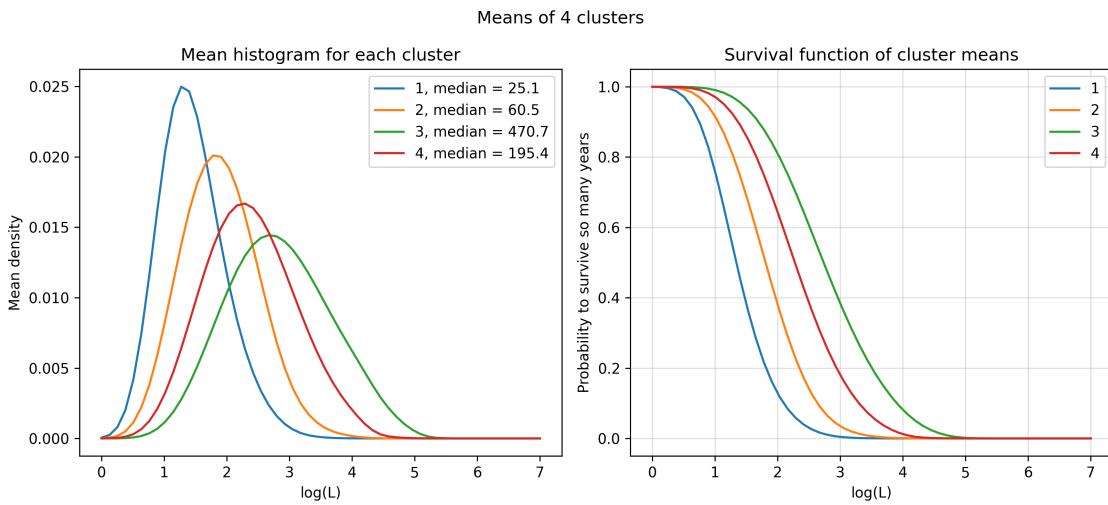
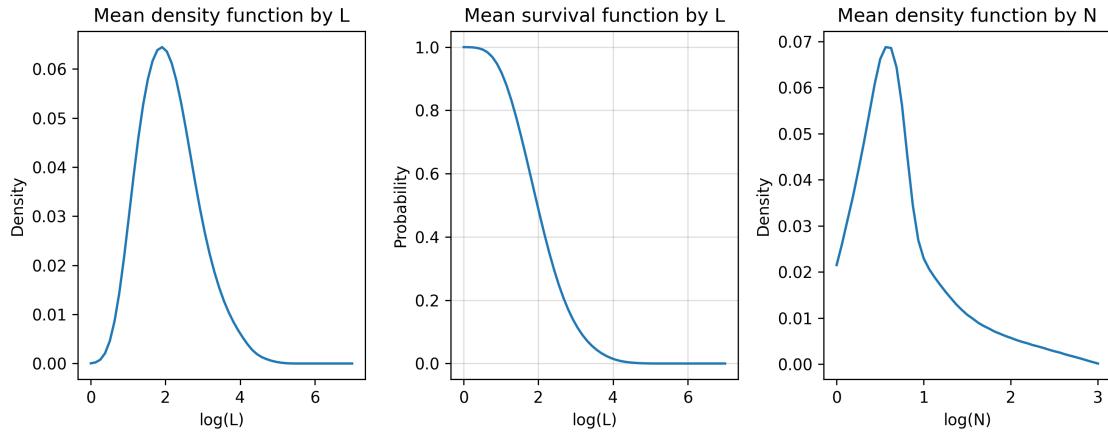


```
[5]: for s in [1, 2]:
    cluster(model=3, ks=[4], supermodel=s) # model 3
```

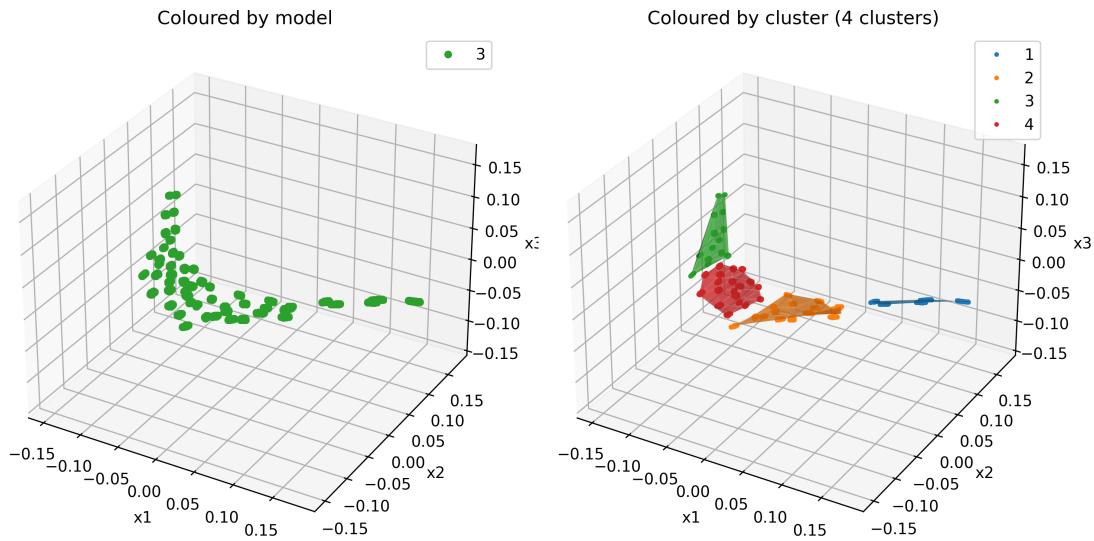
Model III

Mean density

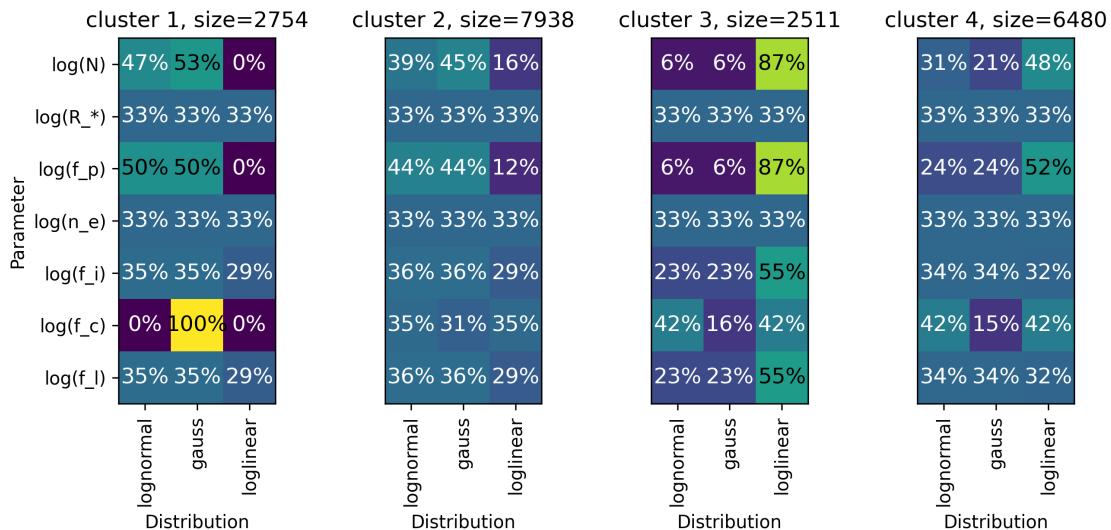




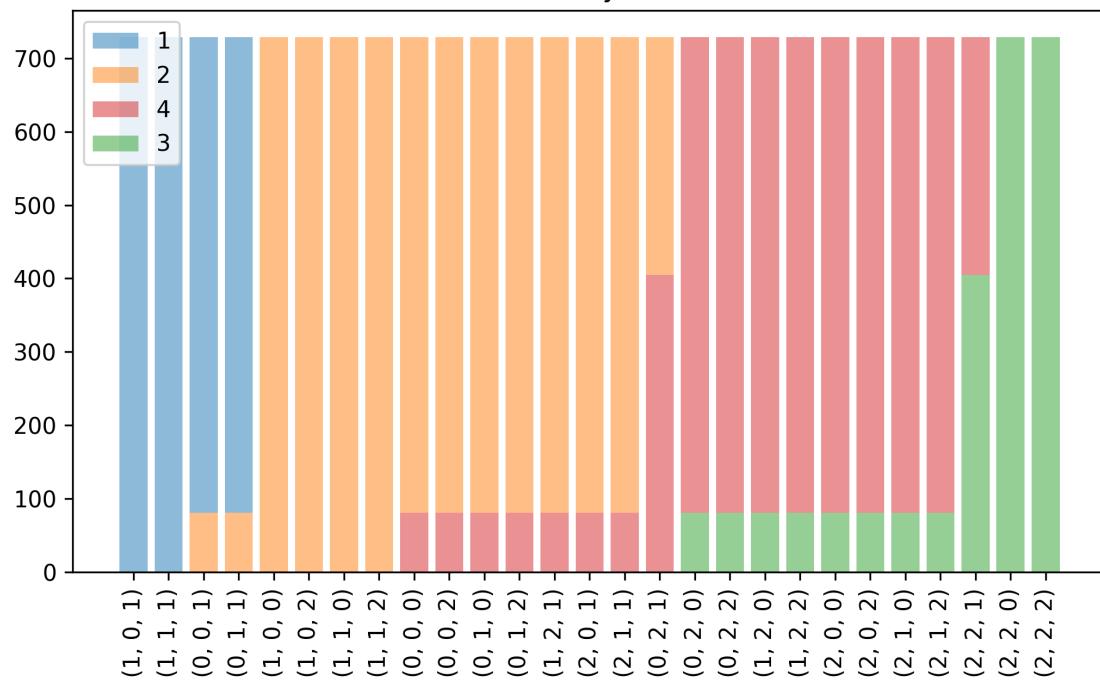
Model III after Principal component analysis (PCA)



Percent of submodels in model 3 distributed with particular distribution on particular parameter in each of 4 clusters

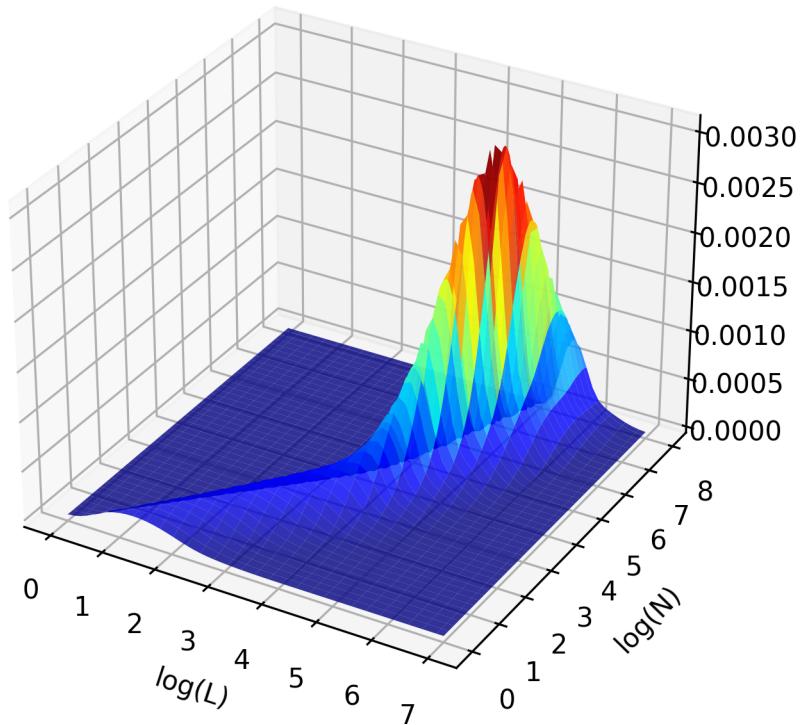


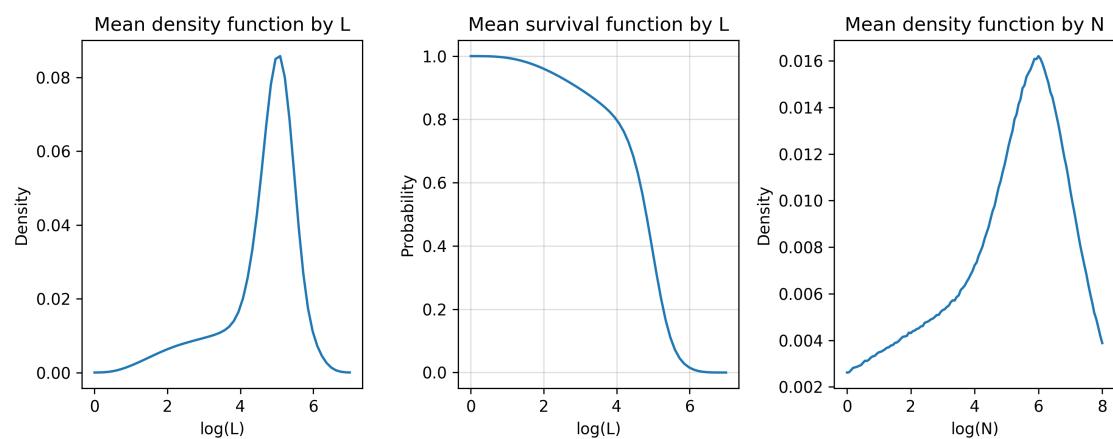
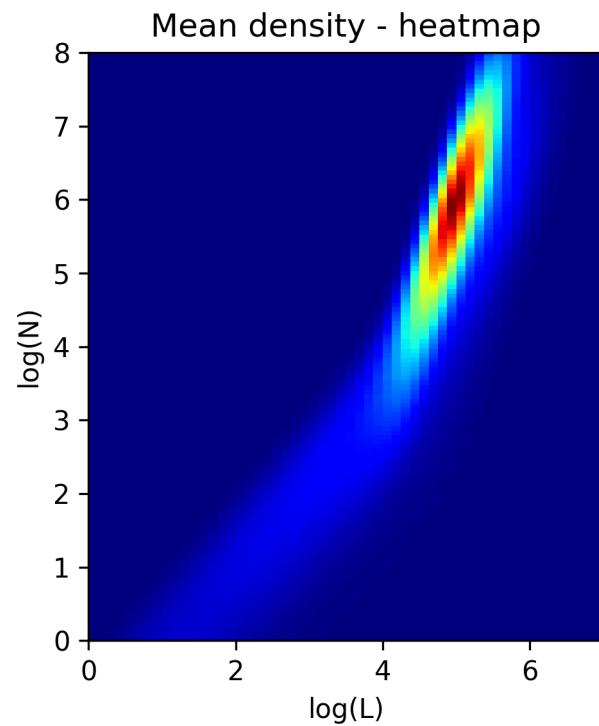
Percentage of distributions
 $(\log(N), \log(f_p), \log(f_c))$ by (lognormal, gauss, loglinear)
 coloured by cluster

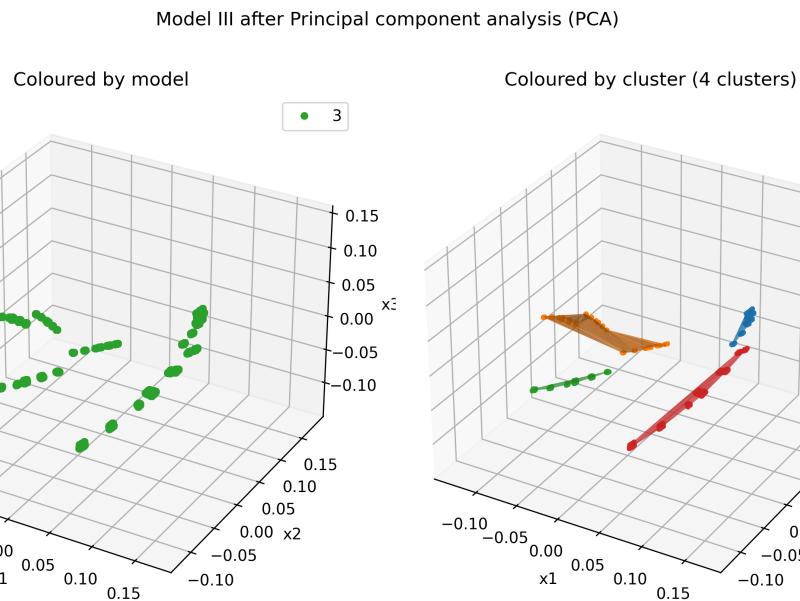
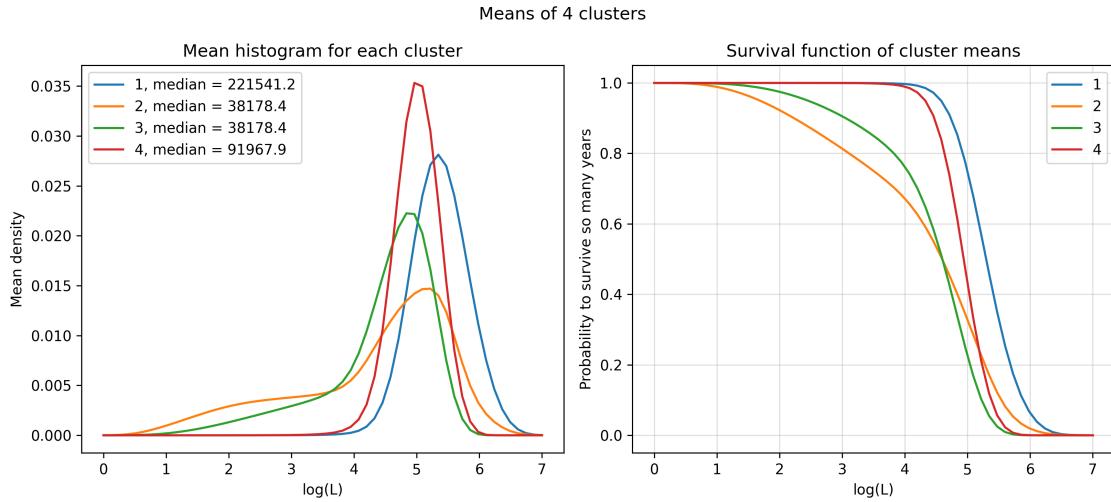


Model III

Mean density



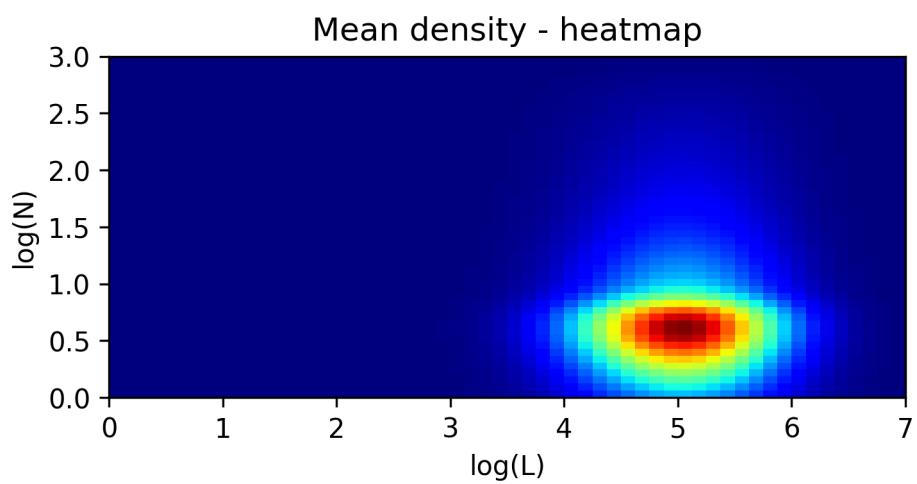
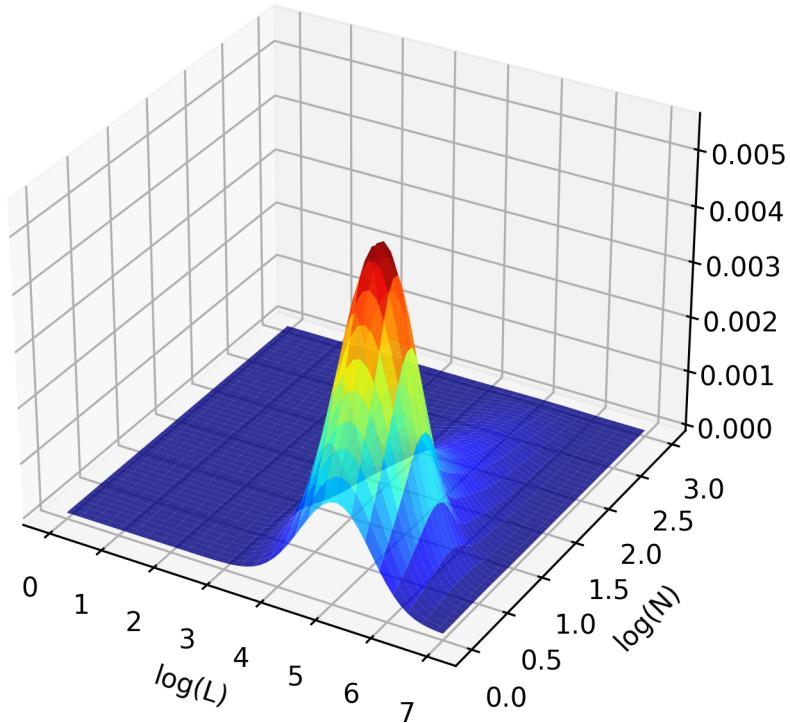


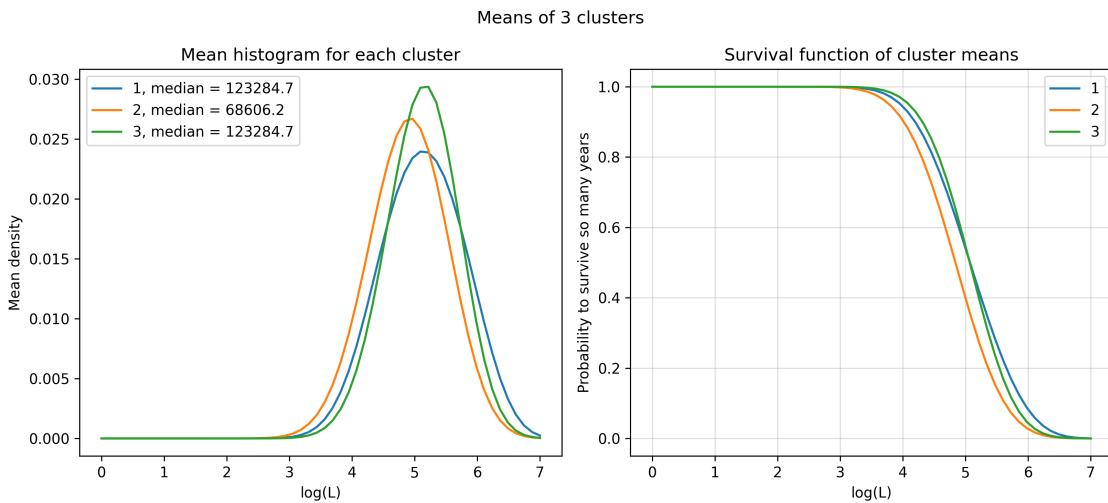
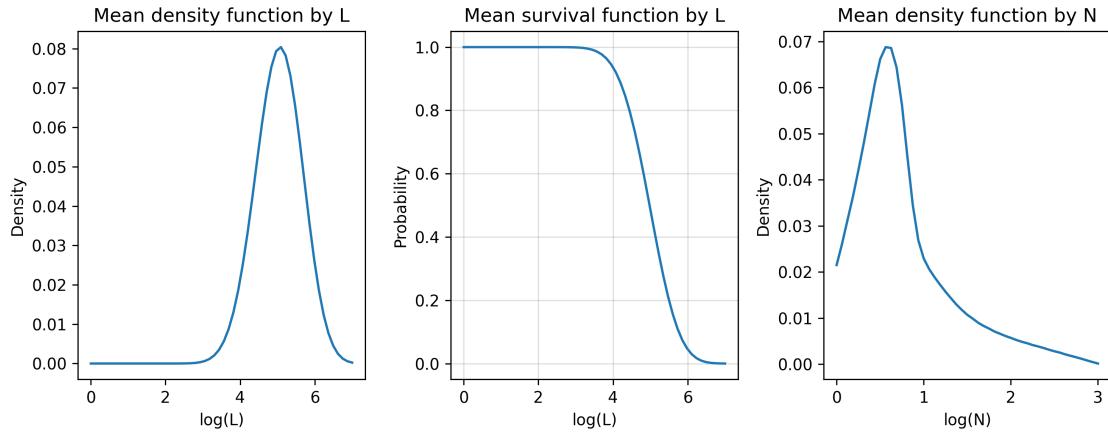


```
[6]: cluster(model=4, ks=[3]) # model 4
```

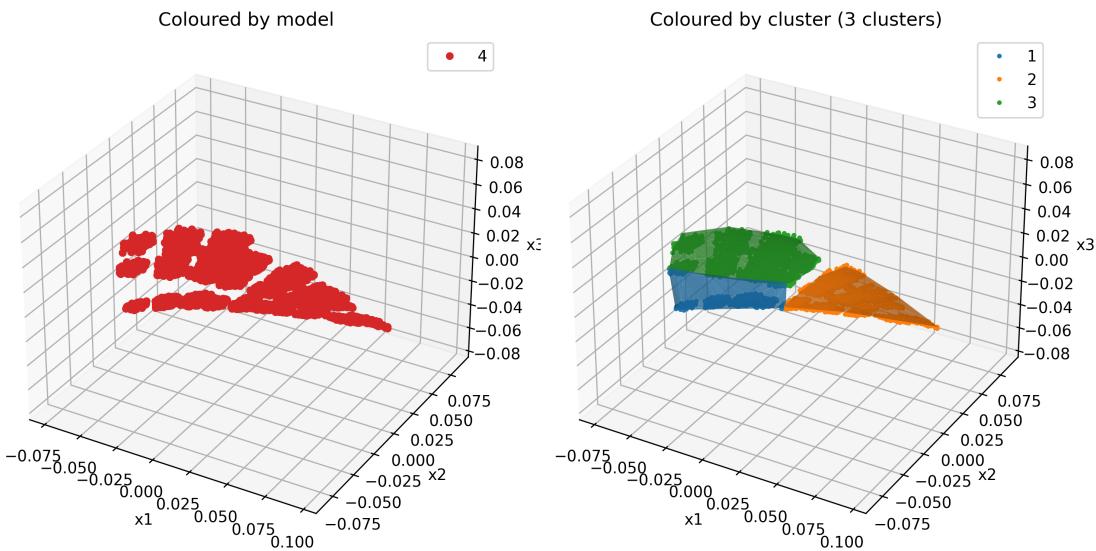
Model IV

Mean density

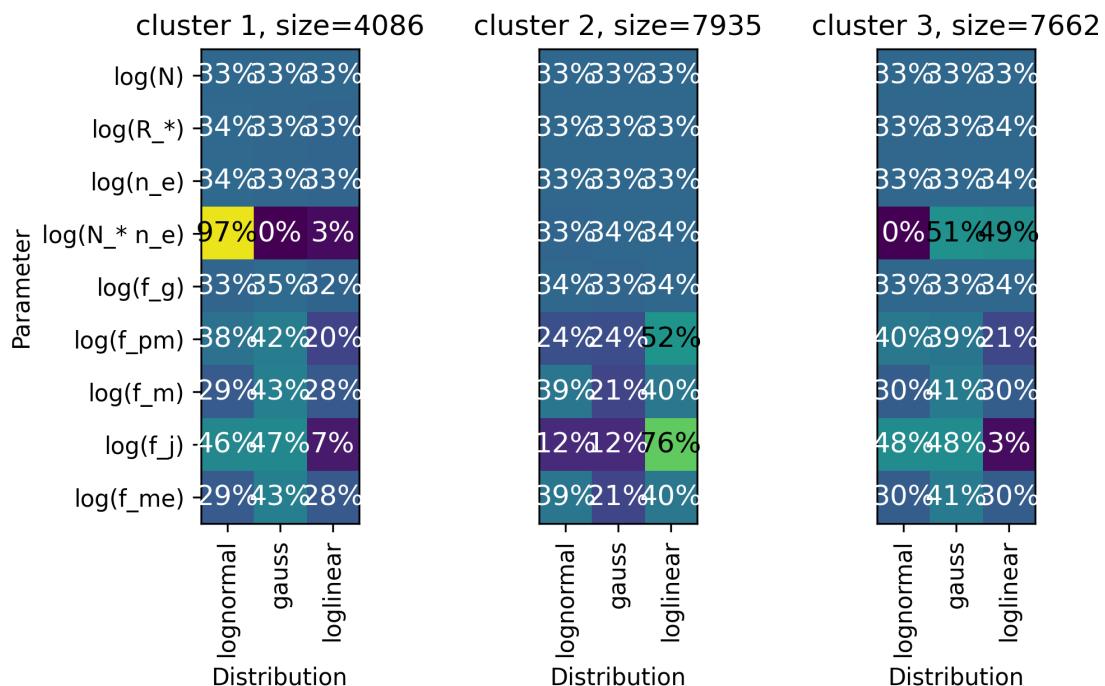




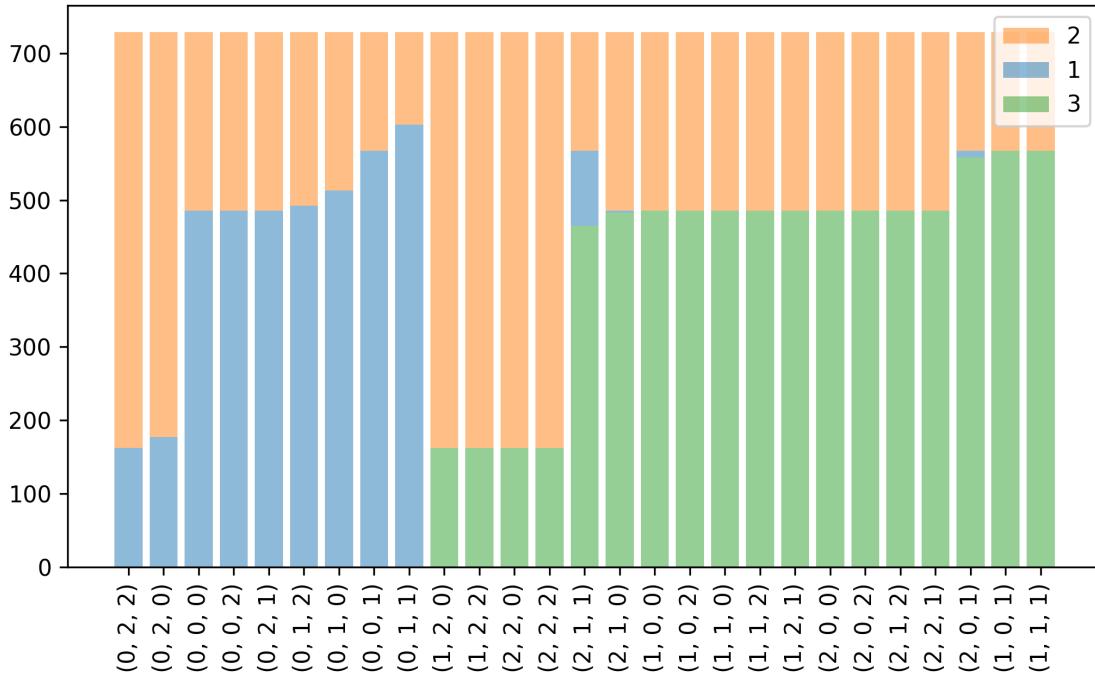
Model IV after Principal component analysis (PCA)



Percent of submodels in model 4 distributed with particular distribution on particular parameter in each of 3 clusters



Percentage of distributions
 $(\log(N \cdot n_e), \log(f_p m), \log(f_m))$ by (lognormal, gauss, loglinear)
 coloured by cluster



```
[7]: def polyDeltaL(logL1):
    minP, meanP, maxP = -2, np.log10(1.8), np.log10(5)
    L1 = 10 ** logL1
    a1 = 10 ** (9 - np.array([minP, meanP, maxP]) - np.log10(np.pi * 4))
    a1 = [np.roots([1, 3 * L1, 3 * L1 ** 2 + a, L1 ** 3]) for a in a1]
    return [- min(a, key=lambda x: np.abs(x.imag)).real for a in a1]

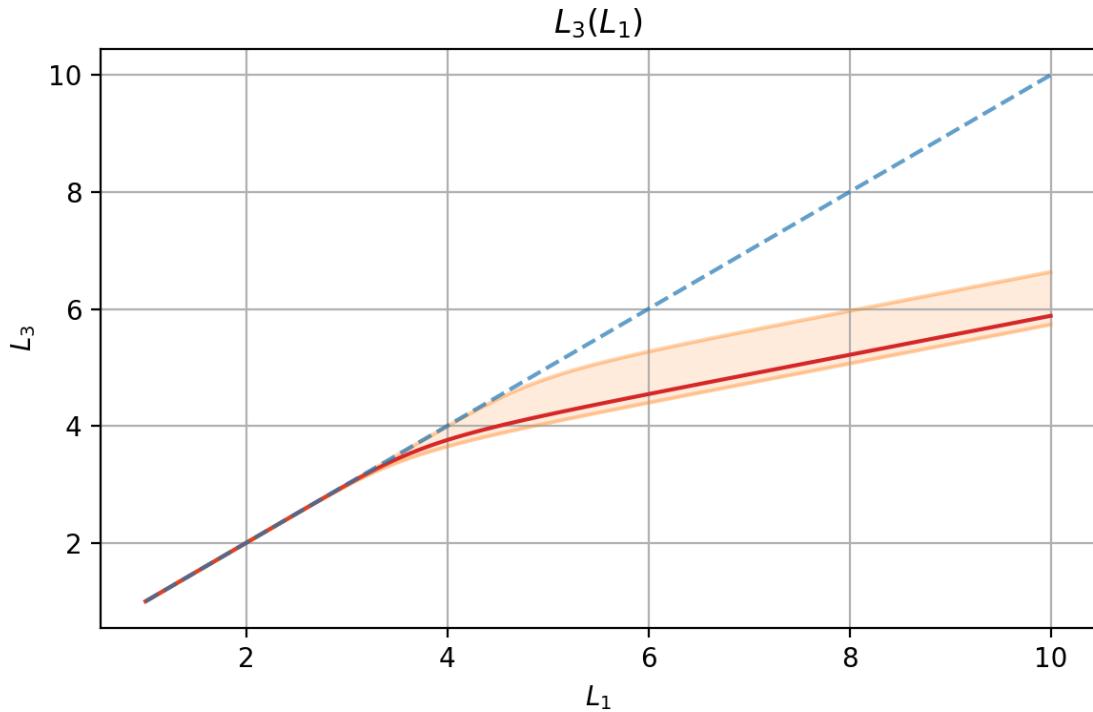
xi = np.linspace(1, 10, 100)
xi3 = np.array([xi, xi, xi]).T
h = np.array([polyDeltaL(x) for x in xi])
ca = [("tab:orange", 0.3), ("tab:red", 1), ("tab:orange", 0.3)]

for ylab, ydat in [("$L_3$", np.log10(np.abs(10 ** xi3 - h)))]:
    plt.figure(figsize=(6, 4), dpi=200, tight_layout=True)
    plt.fill_between(xi, ydat[:, 0], ydat[:, 2],
                     color=ca[0][0], alpha=ca[0][1] / 2)
    for i in range(3):
        plt.plot(xi, ydat[:, i], color=ca[i][0], alpha=ca[i][1])
        if "L_1" not in ylab:
            plt.plot(xi, xi, "--", alpha=0.7)
    plt.title(f"${ylab}({L_1}$")
```

```

plt.xlabel("$L_1$")
plt.ylabel(f"${ylab}$")
plt.grid()
plt.savefig(f"out/model3-model1.png")
plt.show()

```

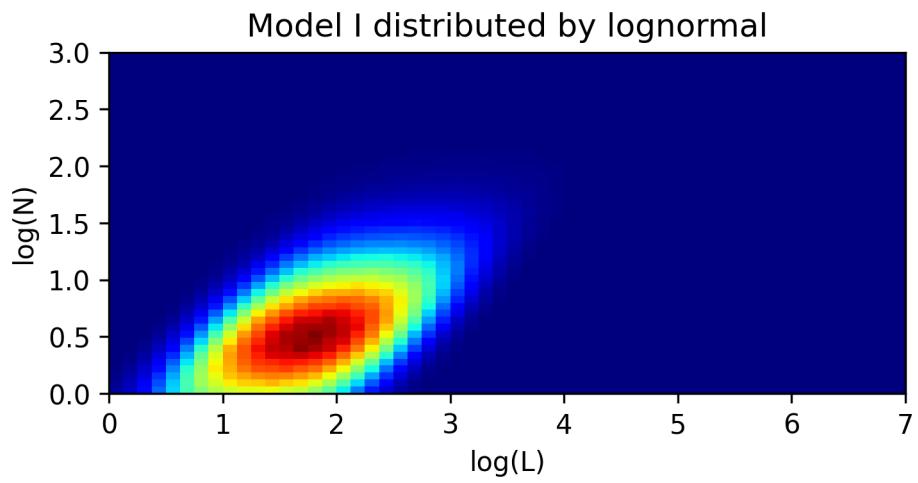
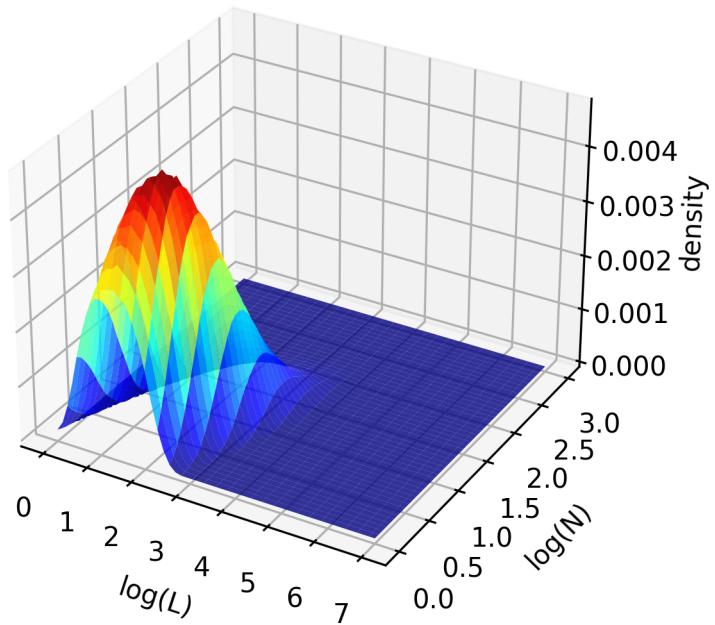


```

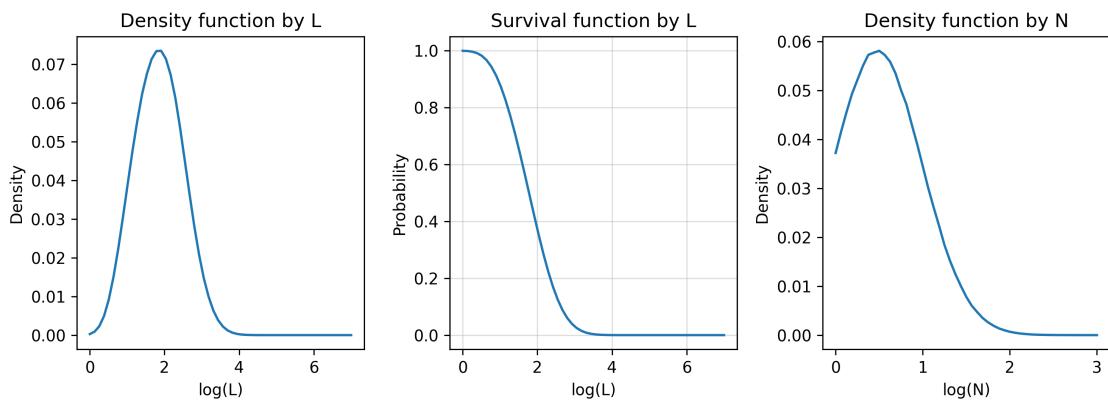
[8]: from plot3D_L import *
draw_histograms3D(model=1, distribution=tuple(0 for _ in range(6)), ↴
    ↵supermodel=1)
draw_histograms3D(model=2, distribution=tuple(0 for _ in range(2)), ↴
    ↵supermodel=1)
draw_histograms3D(model=3, distribution=tuple(1 for _ in range(6)), ↴
    ↵supermodel=1)
draw_histograms3D(model=4, distribution=tuple(0 for _ in range(8)), ↴
    ↵supermodel=1)
draw_histograms_N3D()
plt.show()

```

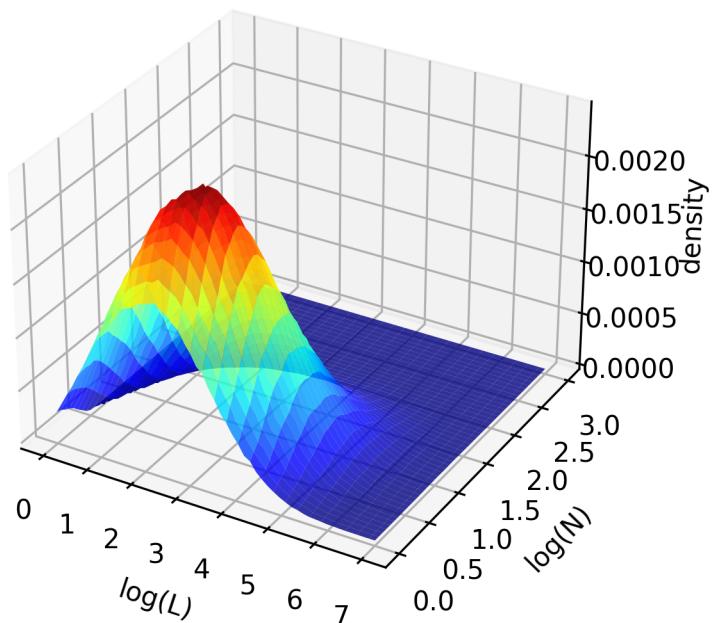
Model I distributed by lognormal



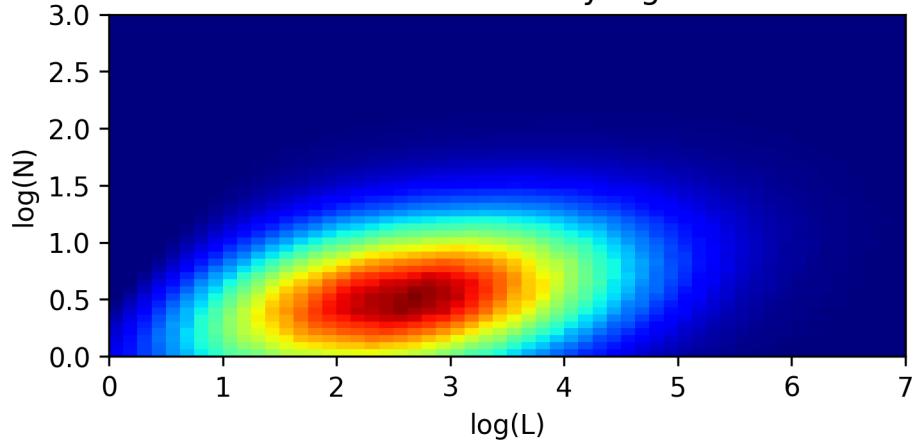
Model I distributed by lognormal



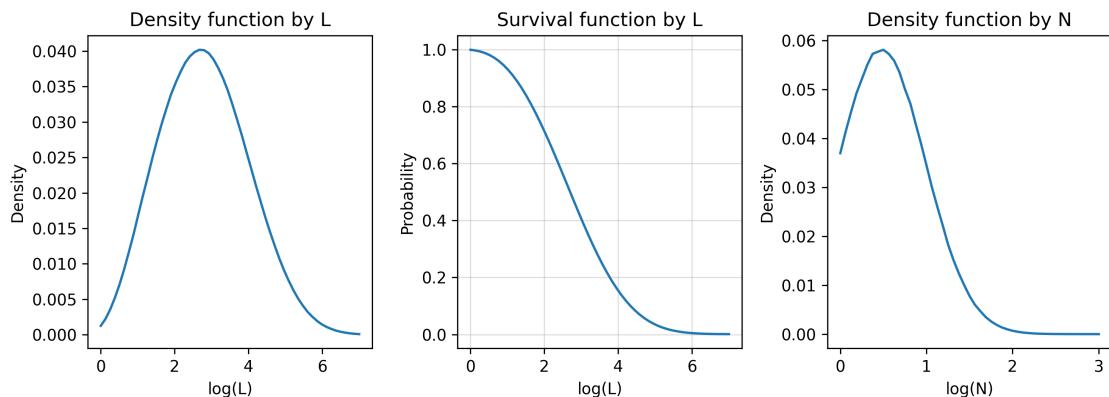
Model II distributed by lognormal



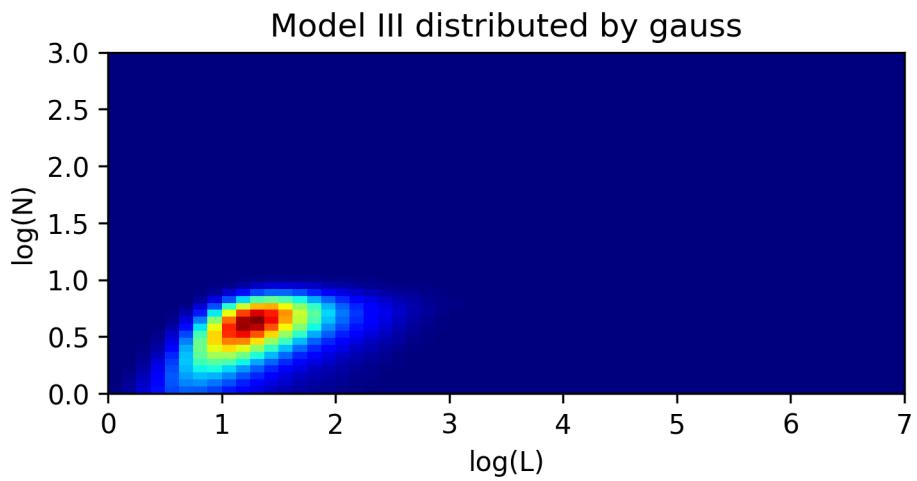
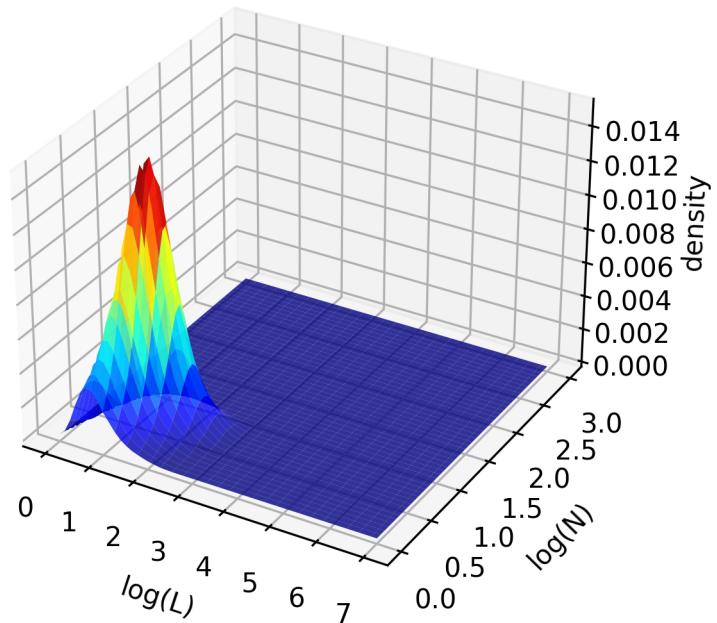
Model II distributed by lognormal



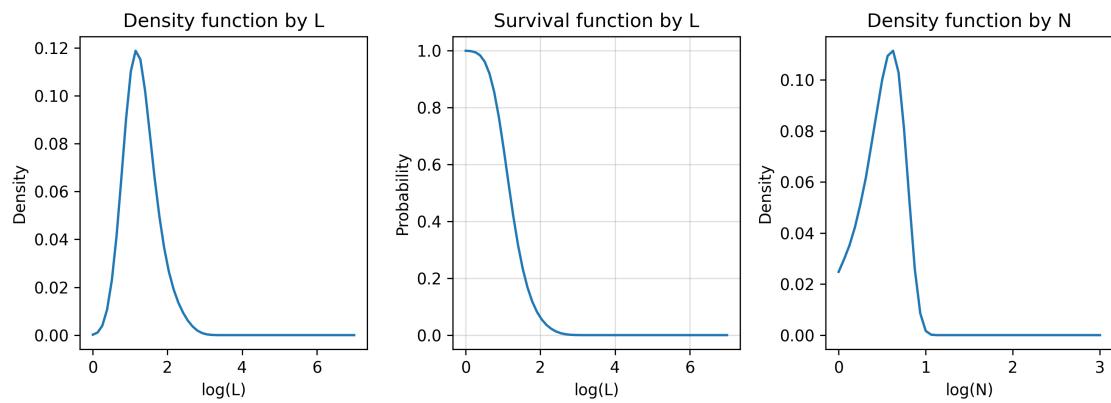
Model II distributed by lognormal



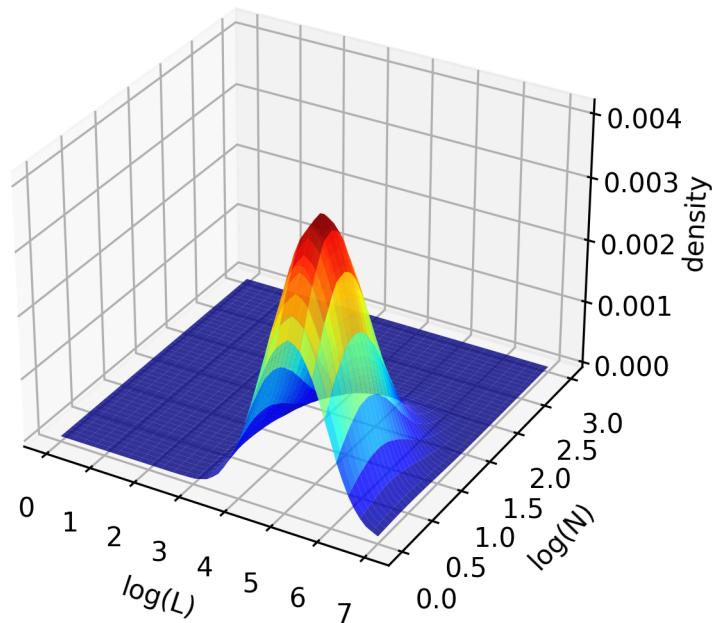
Model III distributed by gauss



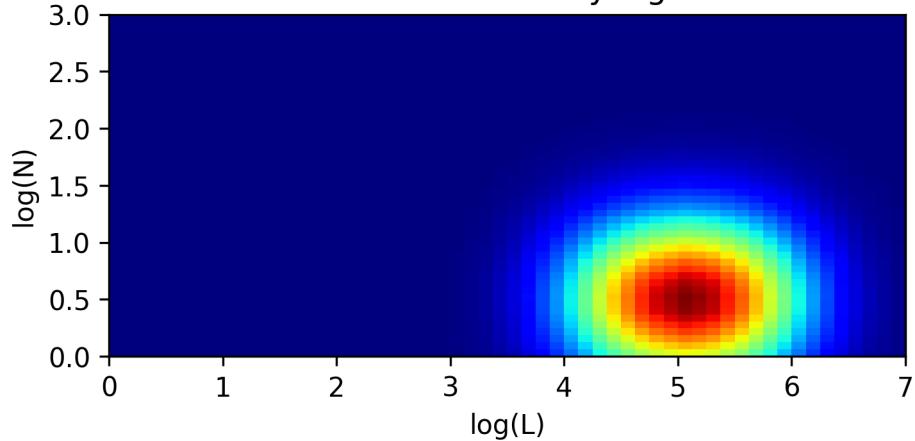
Model III distributed by gauss



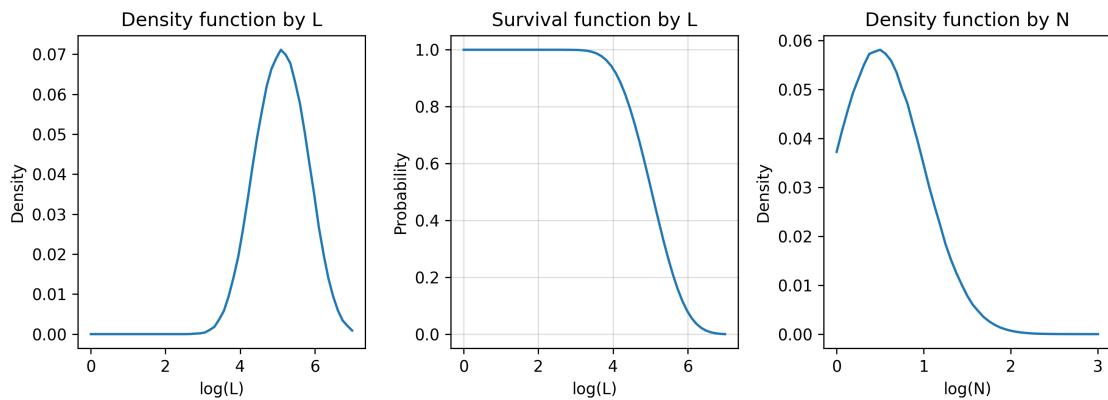
Model IV distributed by lognormal



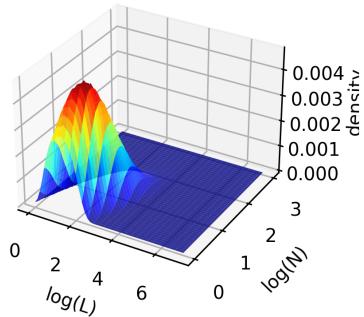
Model IV distributed by lognormal



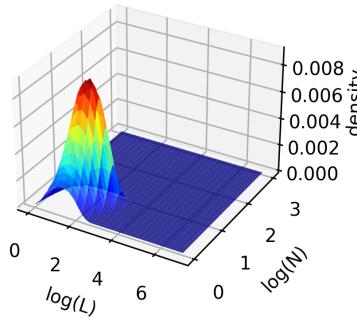
Model IV distributed by lognormal



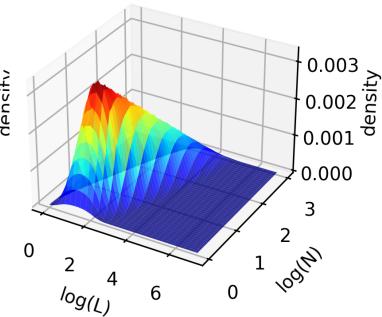
N distributed by lognormal



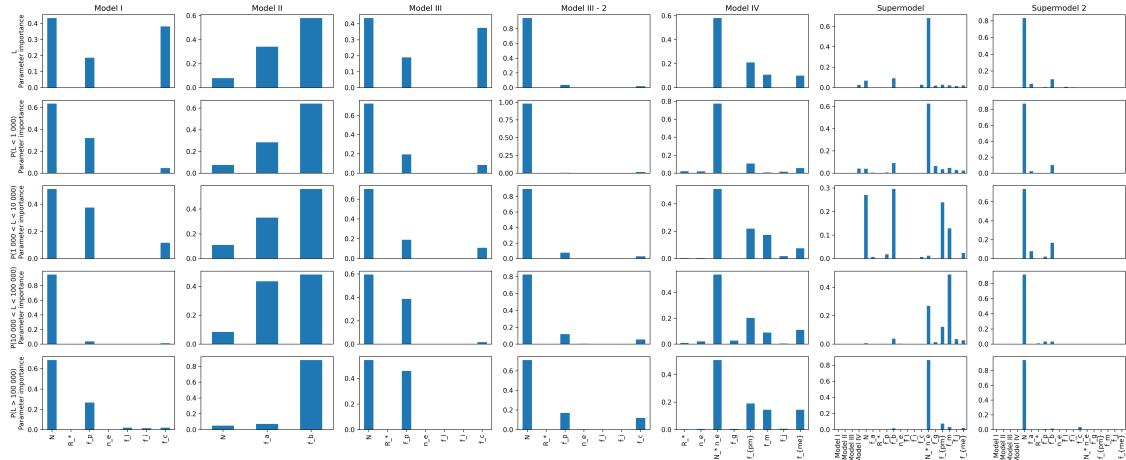
N distributed by gauss



N distributed by loglinear



```
[9]: modeli_napake()
```



```
[10]: data = pd.read_csv('collectedData/meti_tabela_csv.csv', index_col=0)
parameters = list(data.columns)[1:-1]
parameters_nomath = [p.replace("_pm", "_{pm}").replace("_me", "_{me}") if "(" in p else p.replace("_", " ") for p in parameters]
dataTotal = np.load('collectedData/meti_parameters.npy')
labelsTotal = np.load('collectedData/meti_labels.npy')
data = dataTotal[dataTotal[:, 0] == 1, 1:]
labels = labelsTotal[dataTotal[:, 0] == 1]
data2 = dataTotal[dataTotal[:, 0] == 2, 1:]
labels2 = labelsTotal[dataTotal[:, 0] == 2]
columns = [[4, 6, 7, 9, 10, 11, 12], [4, 5, 8], [4, 6, 7, 9, 10, 11, 12], [6, 9, 13, 14, 15, 16, 17, 18]]
podatki = [(data[data[:, 0] == 1, :][:, columns[0]], [parameters[c] for c in columns[0]], labels[data[:, 0] == 1], 'Model I')]
podatki += [(data[data[:, 1] == 1, :][:, columns[1]], [parameters[c] for c in columns[1]], labels[data[:, 1] == 1], 'Model II')]
podatki += [(data[data[:, 2] == 1, :][:, columns[2]], [parameters[c] for c in columns[2]], labels[data[:, 2] == 1], 'Model III')]
podatki += [(data2[data2[:, 2] == 1, :][:, columns[2]], [parameters[c] for c in columns[2]], labels2[data2[:, 2] == 1], 'Model III - 2')]
podatki += [(data[data[:, 3] == 1, :][:, columns[3]], [parameters[c] for c in columns[3]], labels[data[:, 3] == 1], 'Model IV')]
```

```
[11]: for d, p, l, m in podatki:  
    pca = PCA().fit(d)  
    data = pca.transform(d)  
    plt.figure(figsize=(8, 4), dpi=300, tight_layout=True), plt.suptitle(m)  
    plt.subplot(121)
```

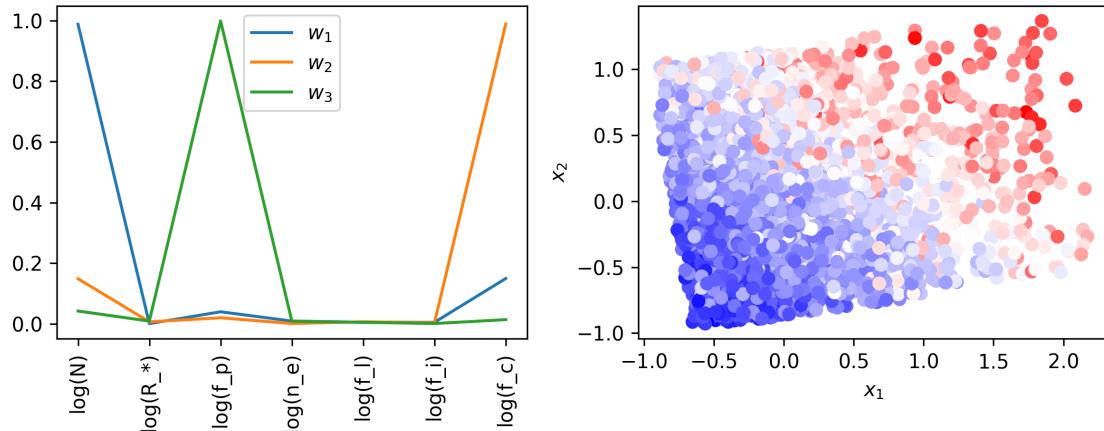
```

plt.plot(np.abs(pca.components_.T[:, 0]), label="$w_1$")
plt.plot(np.abs(pca.components_.T[:, 1]), label="$w_2$")
plt.plot(np.abs(pca.components_.T[:, 2]), label="$w_3$")
plt.xticks(list(range(len(p))), p, rotation=90)
plt.title(f"Weights, {list(np.round(pca.explained_variance_[:3] / np.
    sum(pca.explained_variance_) * 100, 1))}" + f"\nknowledge = {(np.sum(pca.explained_variance_[:3]) / np.
    sum(pca.explained_variance_)) * 100:.1f} %")
plt.legend(loc="best")
ax = plt.subplot(122)
ax.scatter(data[:, 0], data[:, 1], c=l, cmap="bwr")
plt.xlabel("$x_1$"), plt.ylabel("$x_2$")
plt.show()

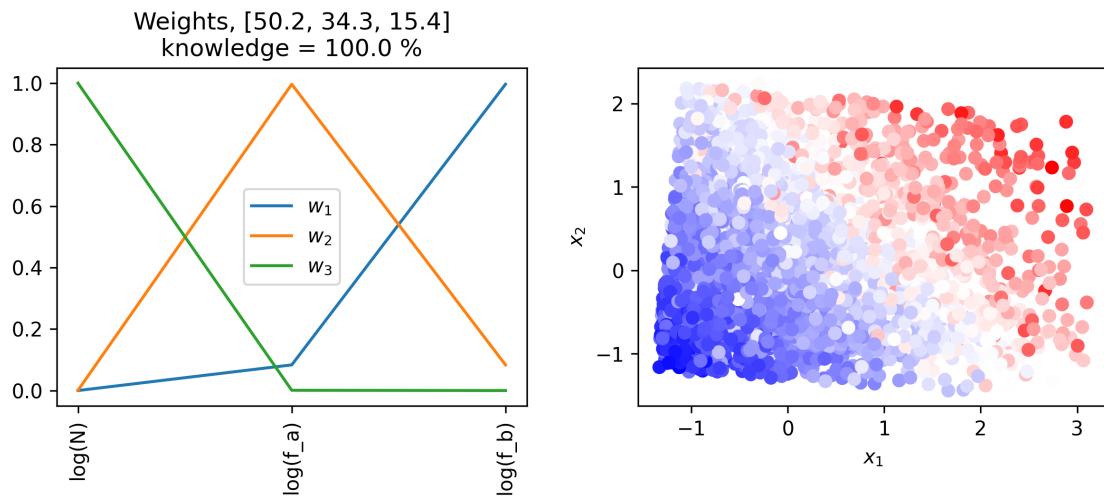
```

Model I

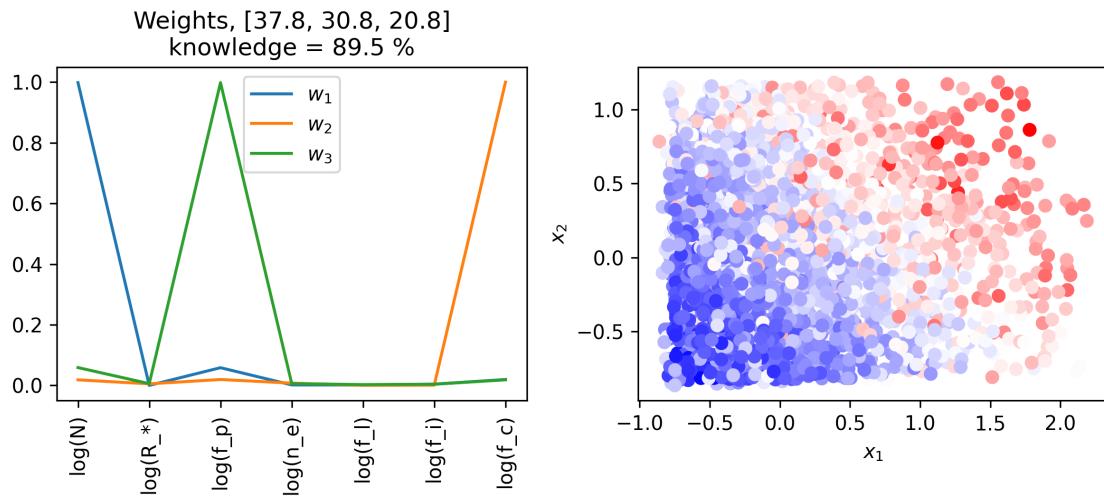
Weights, [36.6, 32.0, 20.6]
knowledge = 89.2 %



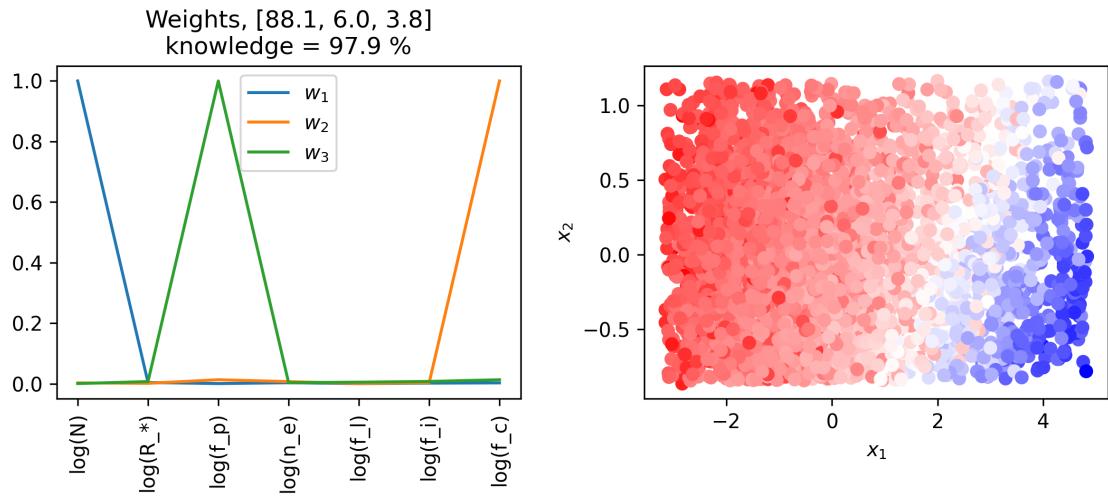
Model II



Model III



Model III - 2



Model IV

