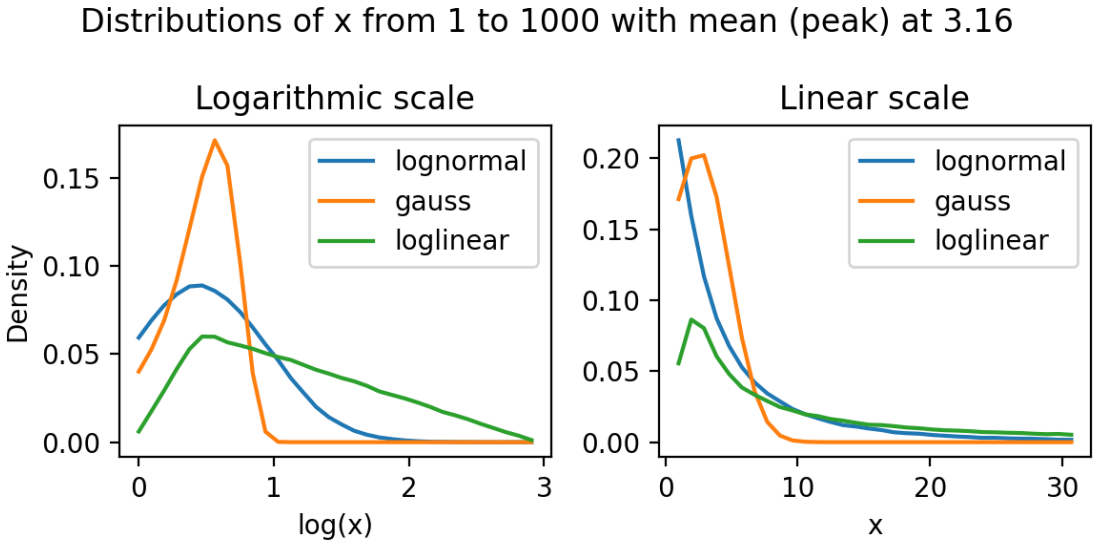
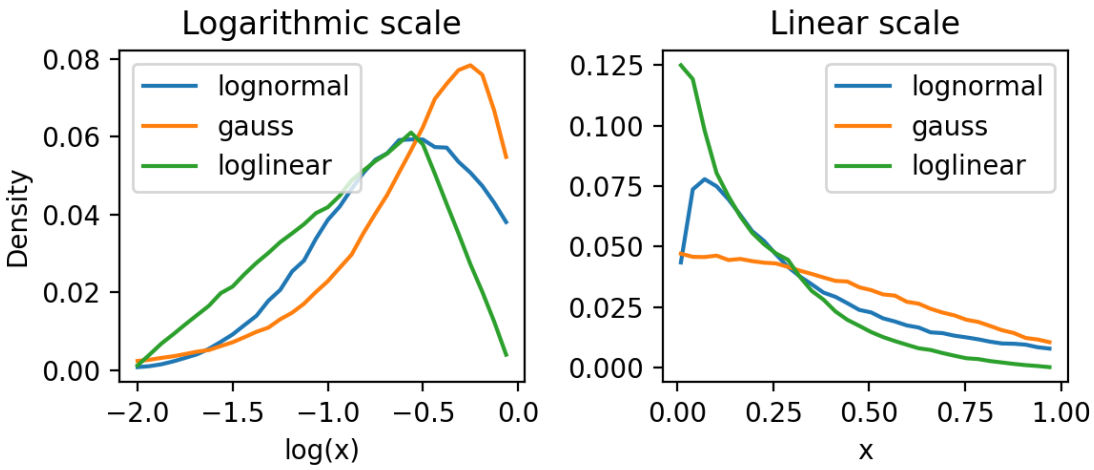
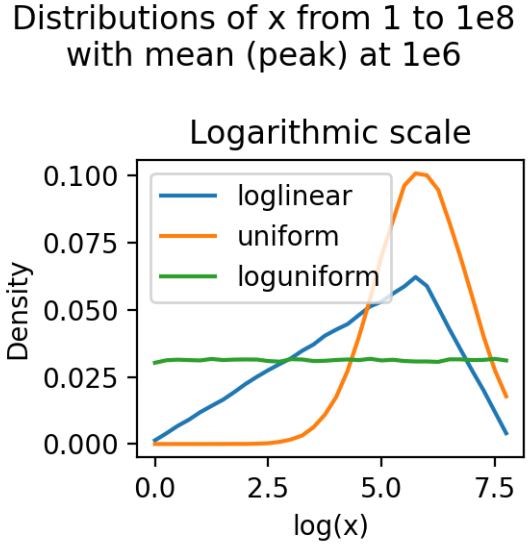
supermodel

January 12, 2022

[1]:
```
from pca import cluster
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import warnings
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
```

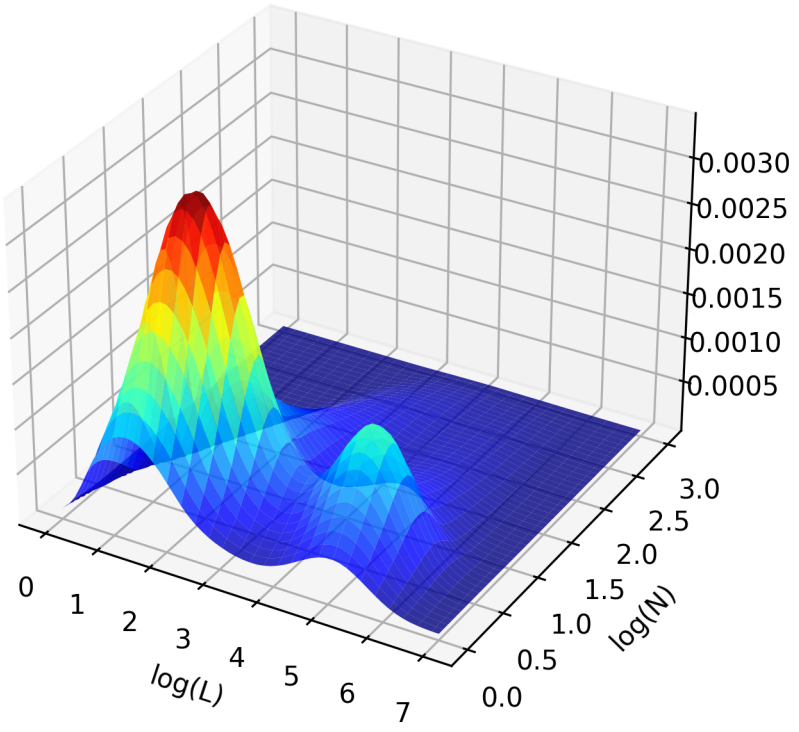


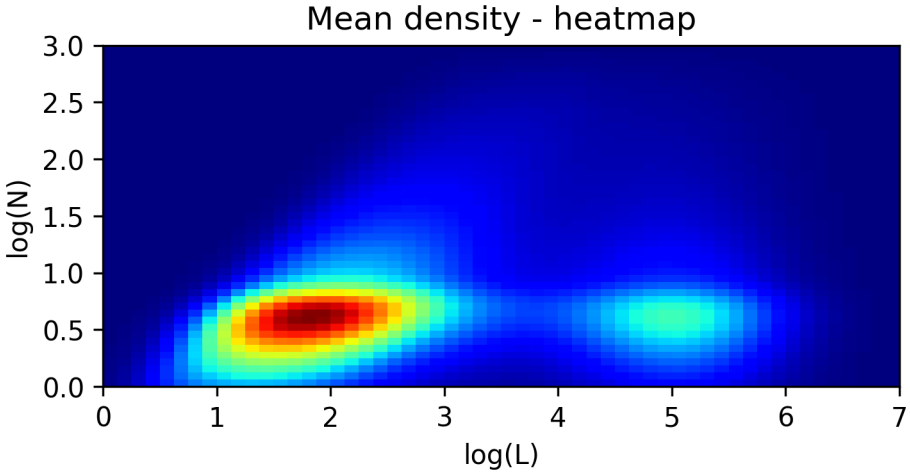


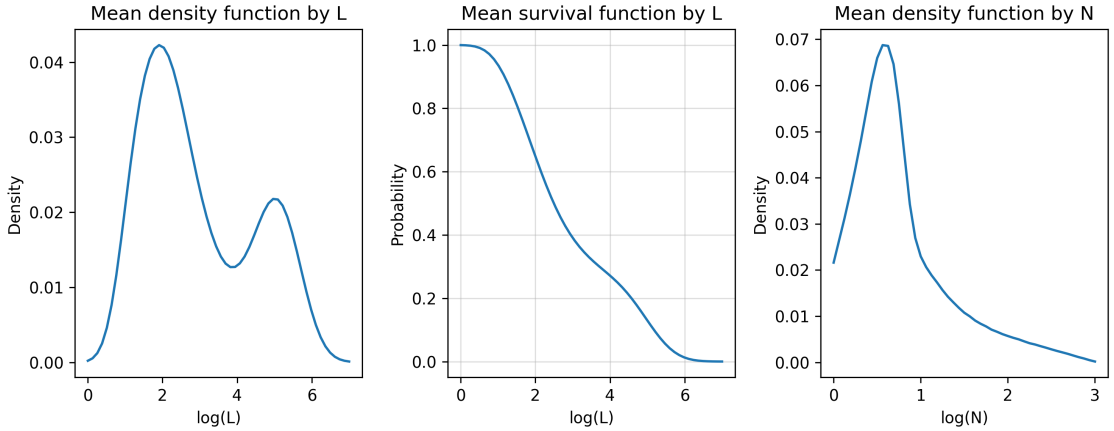


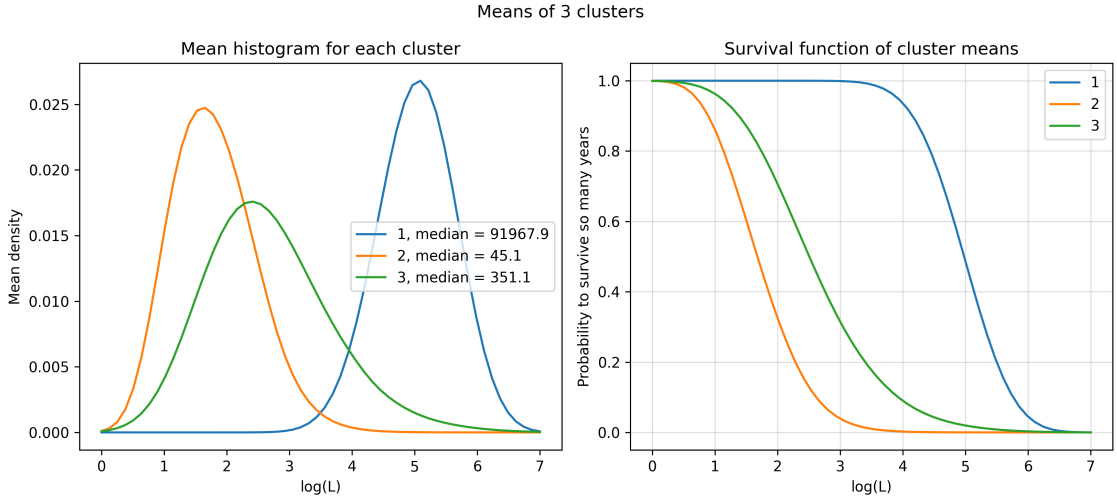
[2]:
```
cluster(model=0, ks=[3, 5], supermodel=1)  # supermodel
cluster(model=0, ks=[5], supermodel=2)  # supermodel 2
```

Supermodel

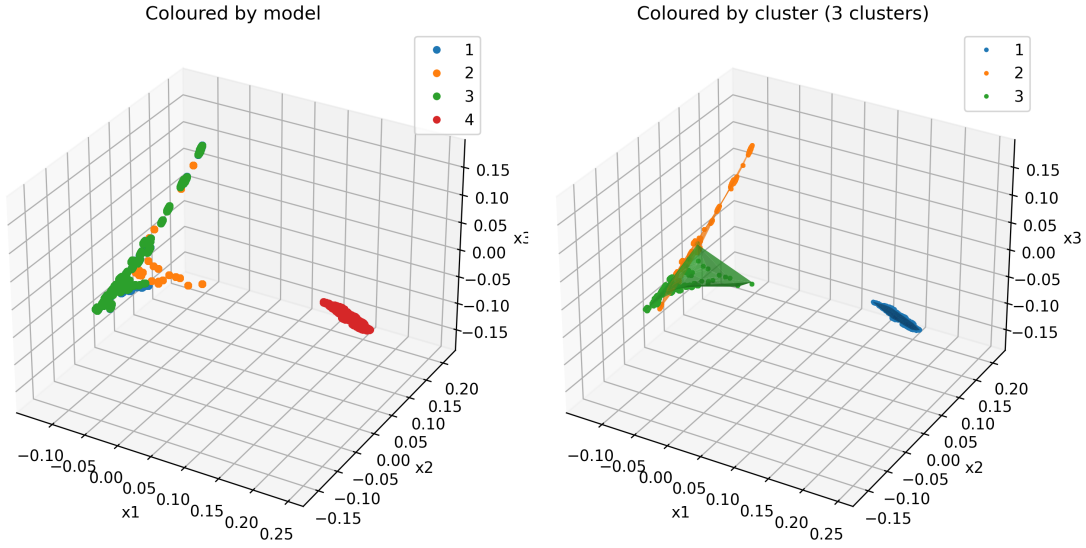

Mean density
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0
1
2
3
4
5
6
7
log(L)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
log(N)


Mean density - heatmap
3.0
2.5
2.0
1.5
1.0
0.5
0.0
log(N)
0
1
2
3
4
5
6
7
log(L)

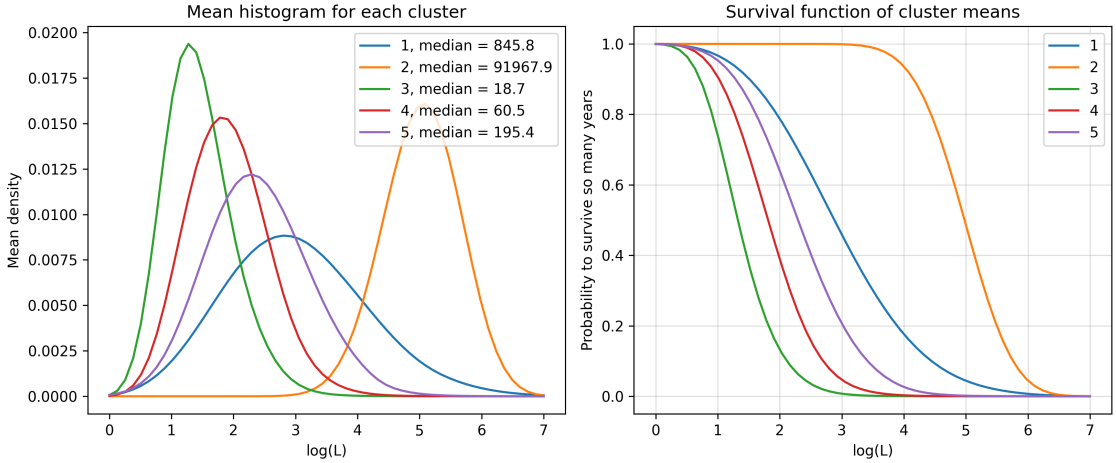

Mean density function by L
Density
log(L)
Mean survival function by L
Probability
log(L)
Mean density function by N
Density
log(N)

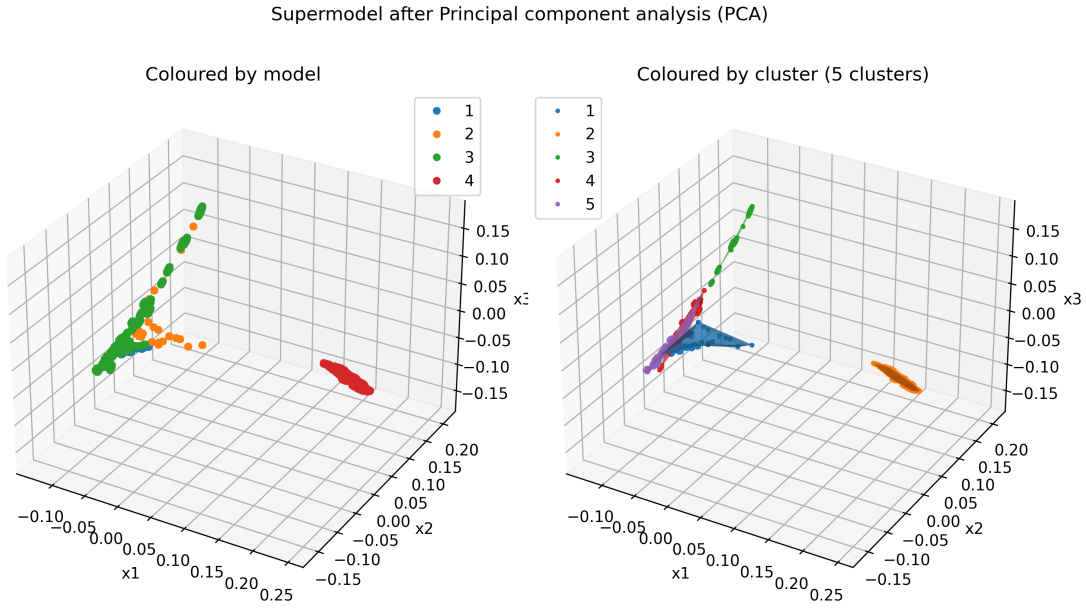

Means of 3 clusters
Mean histogram for each cluster
Mean density
log(L)
1, median = 91967.9
2, median = 45.1
3, median = 351.1
Survival function of cluster means
Probability to survive so many years
log(L)
1
2
3

Supermodel after Principal component analysis (PCA)



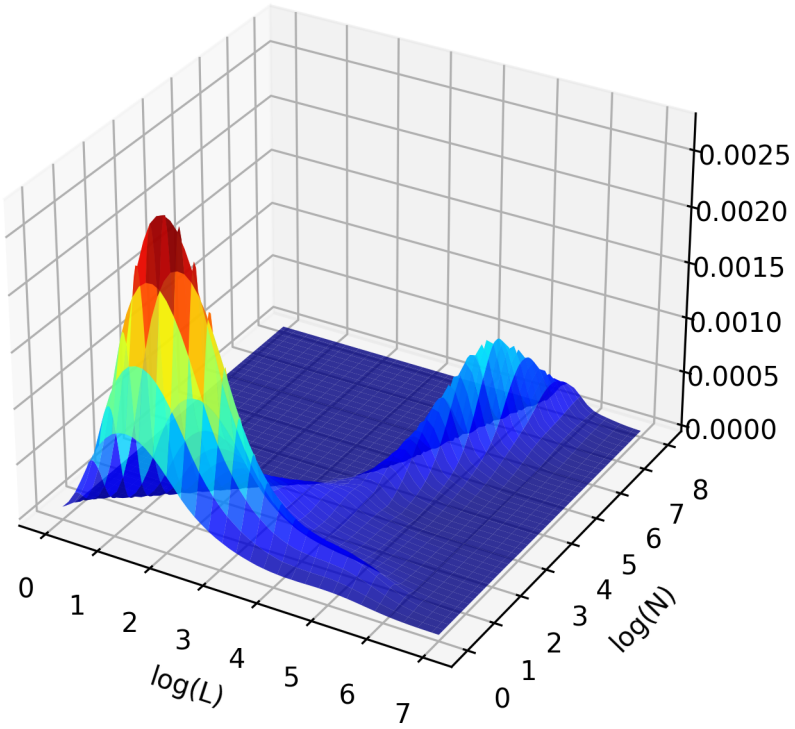
Means of 5 clusters

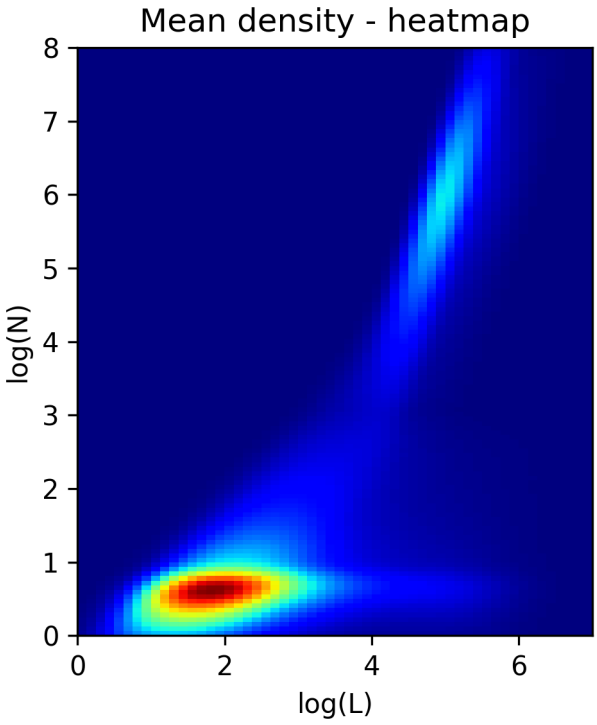


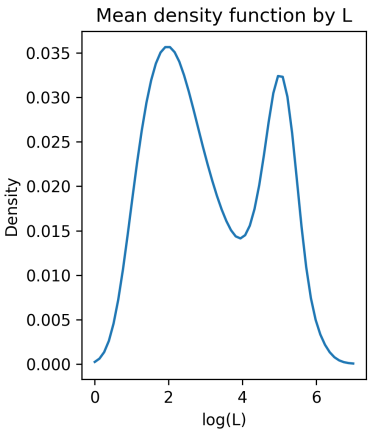


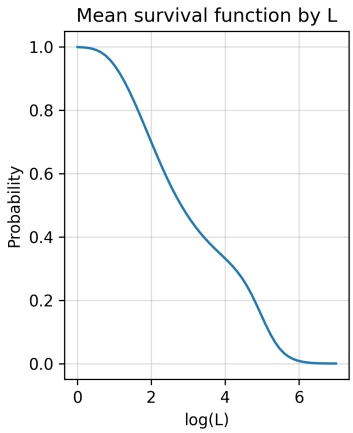
Supermodel 2

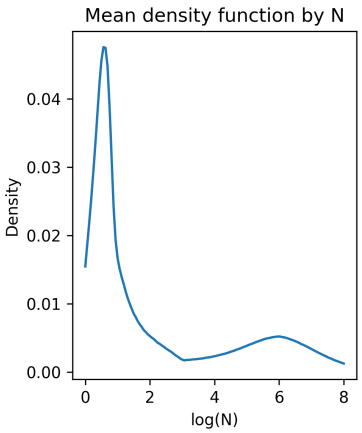
Mean density

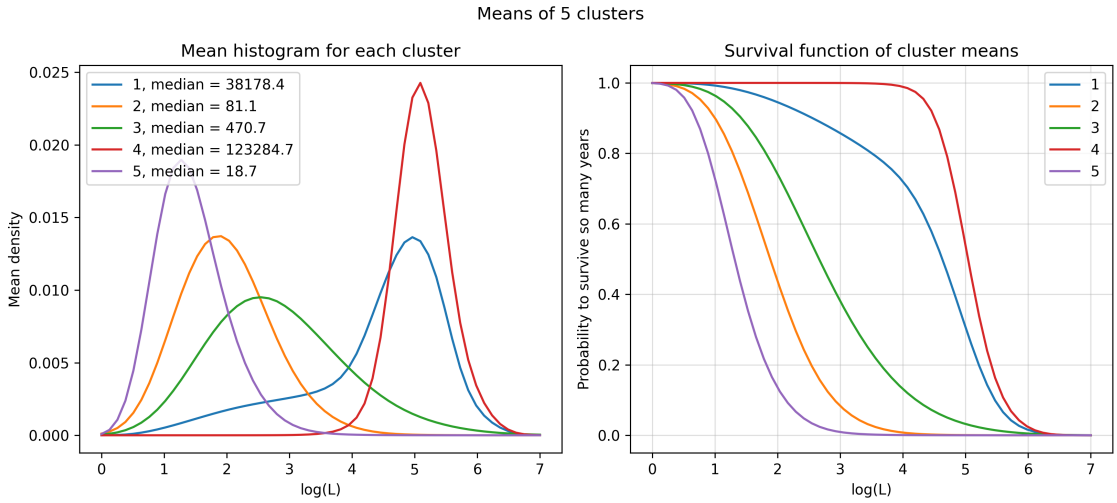


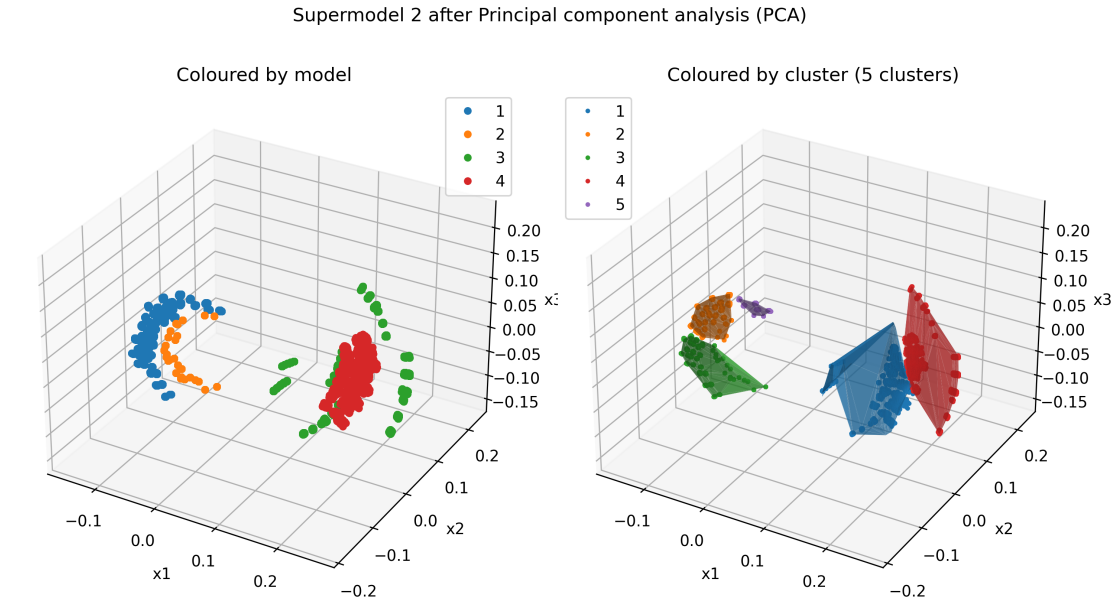

Mean density - heatmap
log(N)
log(L)
0
1
2
3
4
5
6
7
8
0
2
4
6


Mean density function by L
Density
log(L)
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
0
2
4
6


Mean survival function by L
Probability
log(L)
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6


Mean density function by N
Density
log(N)
0.00
0.01
0.02
0.03
0.04
0
2
4
6
8

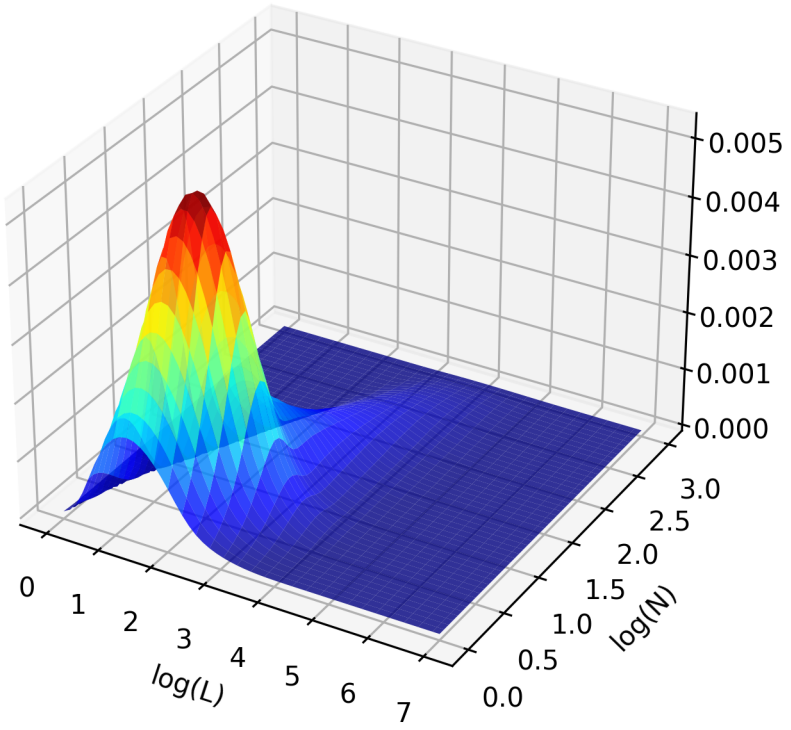




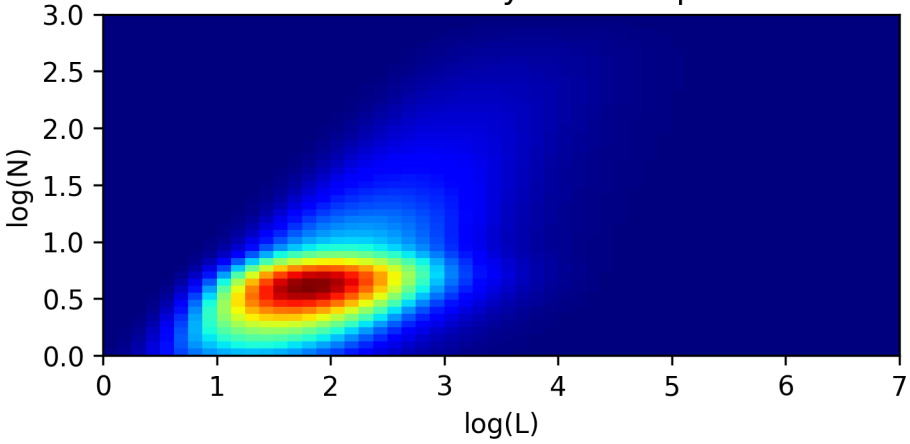
[3]:
```
cluster(model=1, ks=[4])  # model 1
```

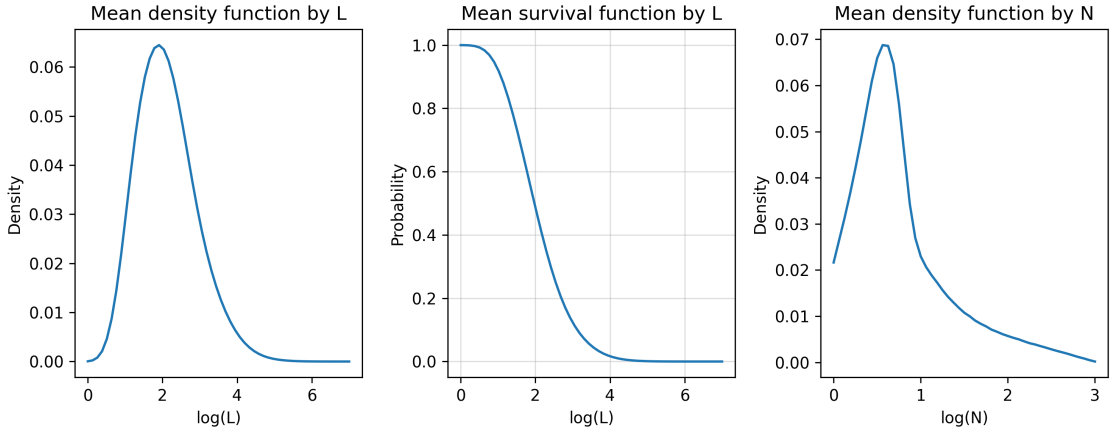
Model I

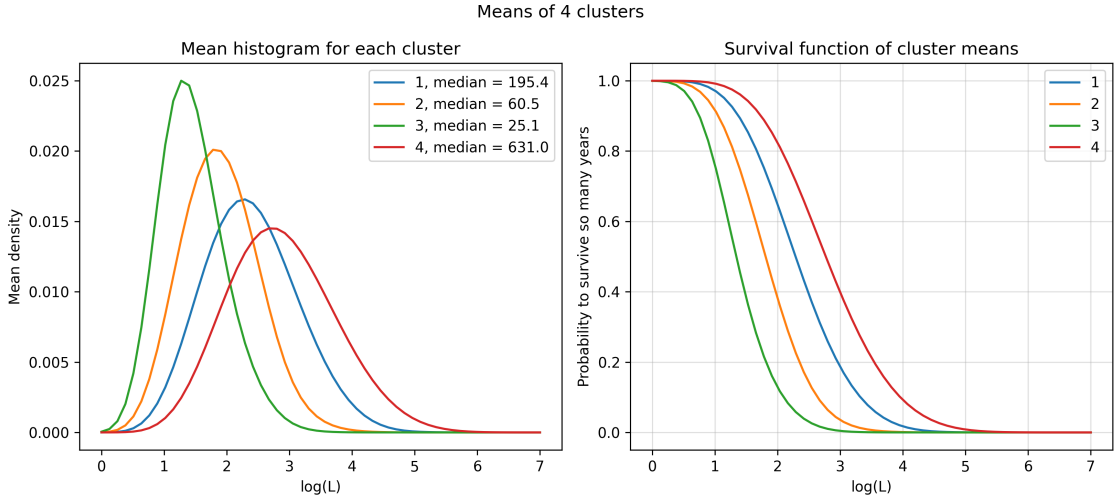
Mean density



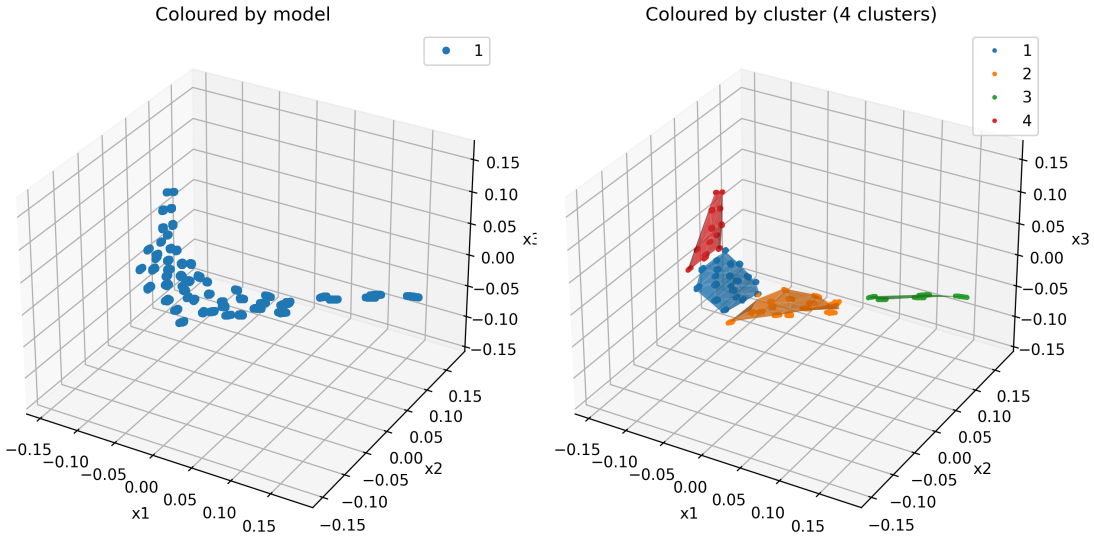
Mean density - heatmap



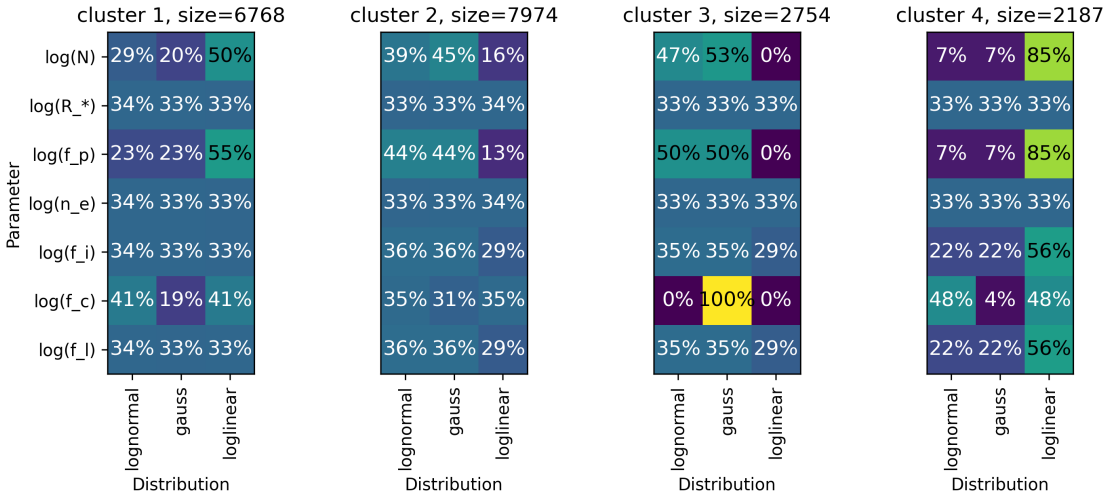

Mean density function by L
Density
log(L)
Mean survival function by L
Probability
log(L)
Mean density function by N
Density
log(N)

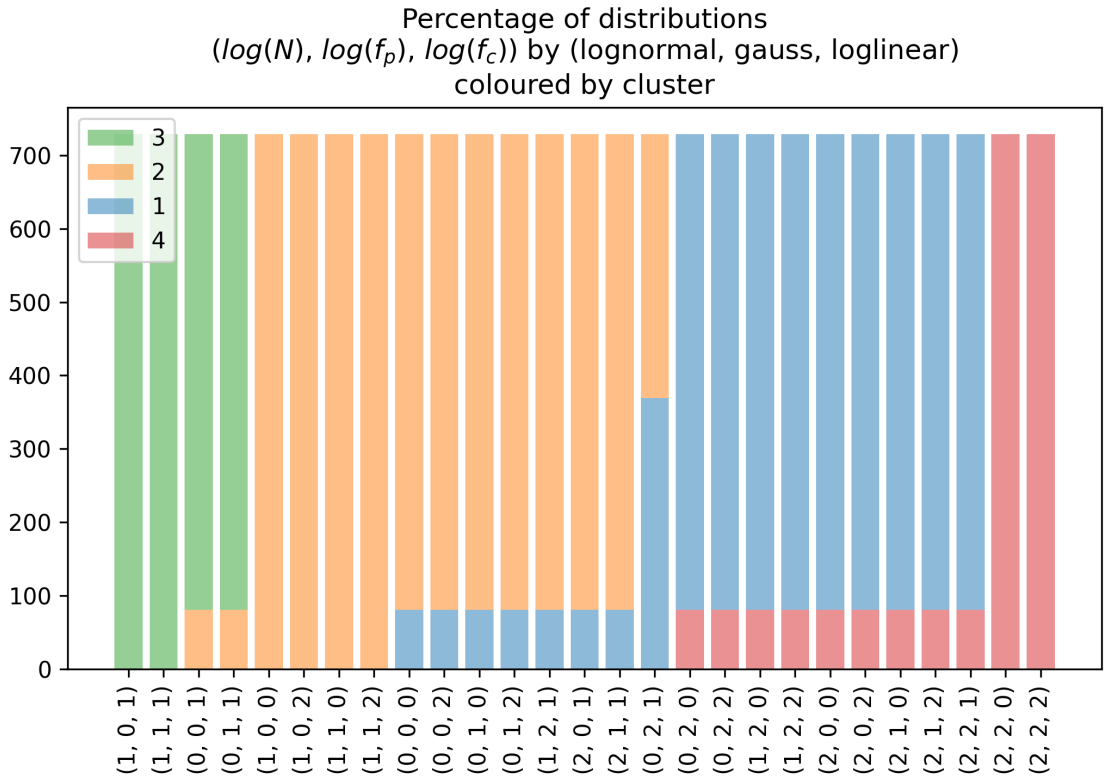

Means of 4 clusters
Mean histogram for each cluster
Mean density
log(L)
1, median = 195.4
2, median = 60.5
3, median = 25.1
4, median = 631.0
Survival function of cluster means
Probability to survive so many years
log(L)
1
2
3
4

Model I after Principal component analysis (PCA)



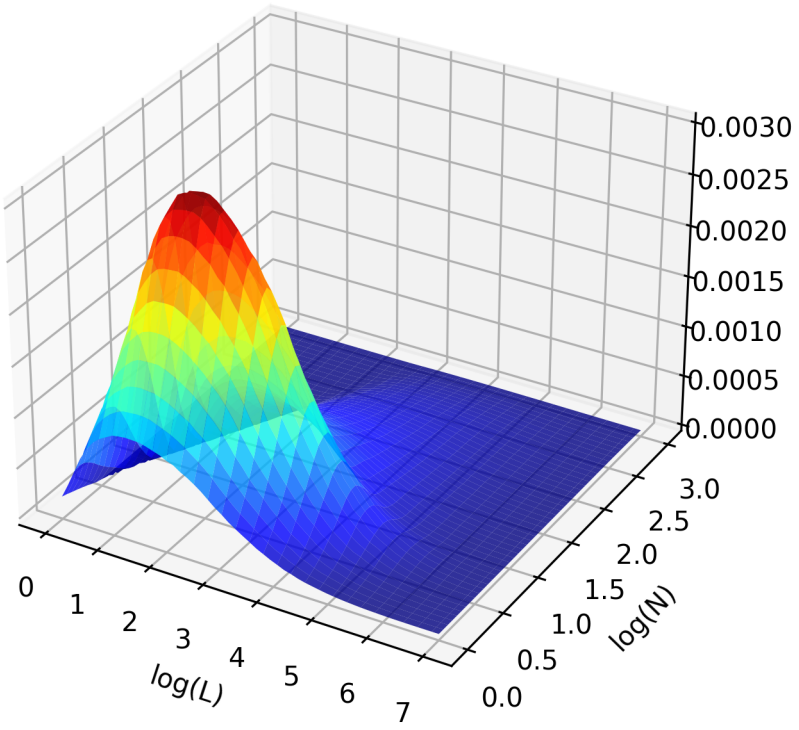
Percent of submodels in model 1 distributed with particular distribution on particular parameter in each of 4 clusters

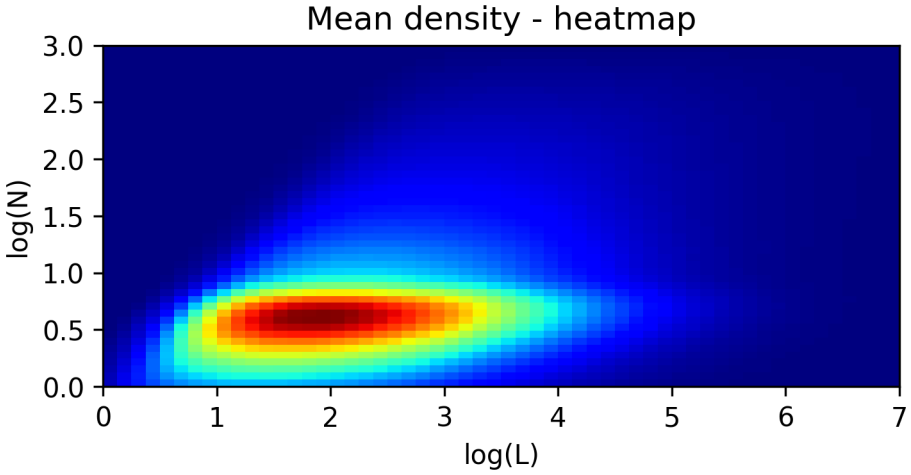


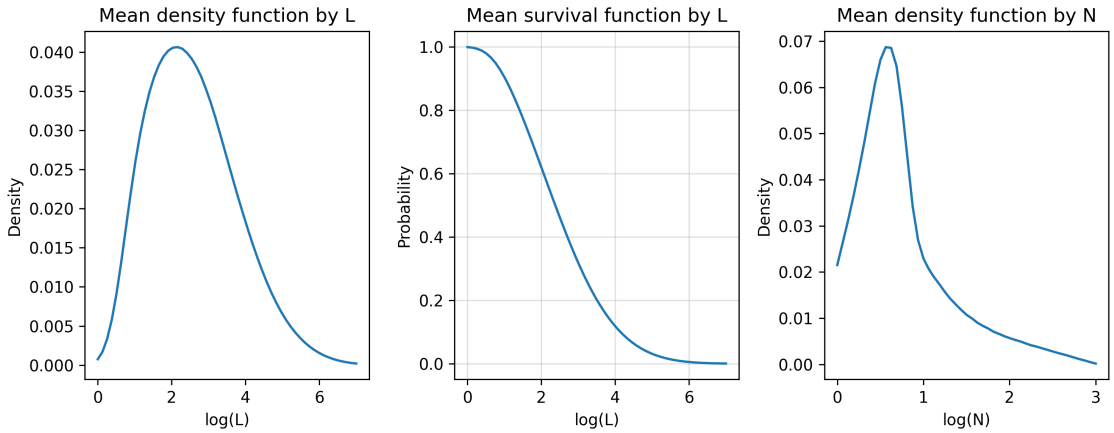


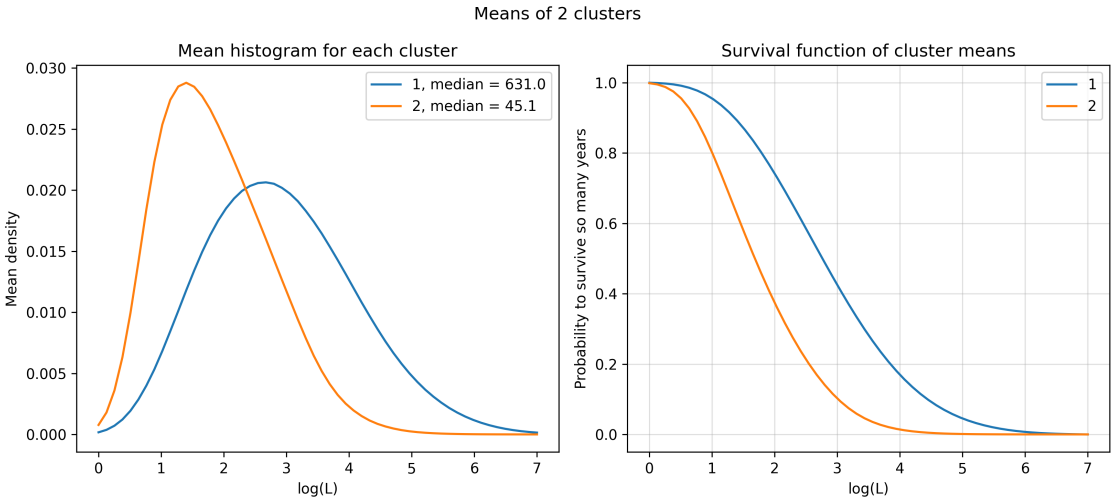
[4]:
```
cluster(model=2, ks=[2])  # model 2
```

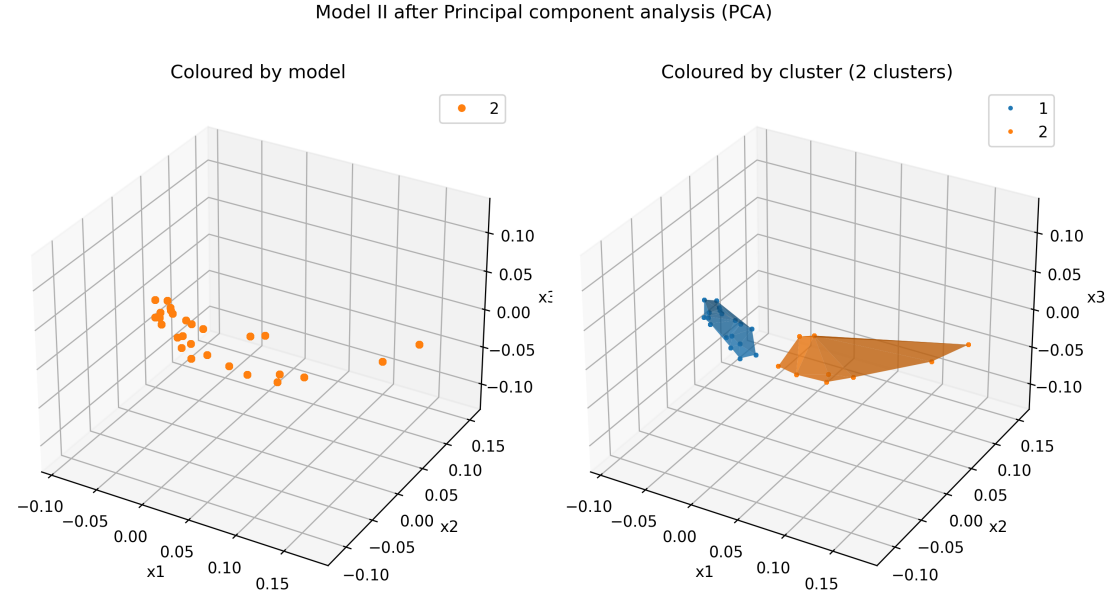
Model II

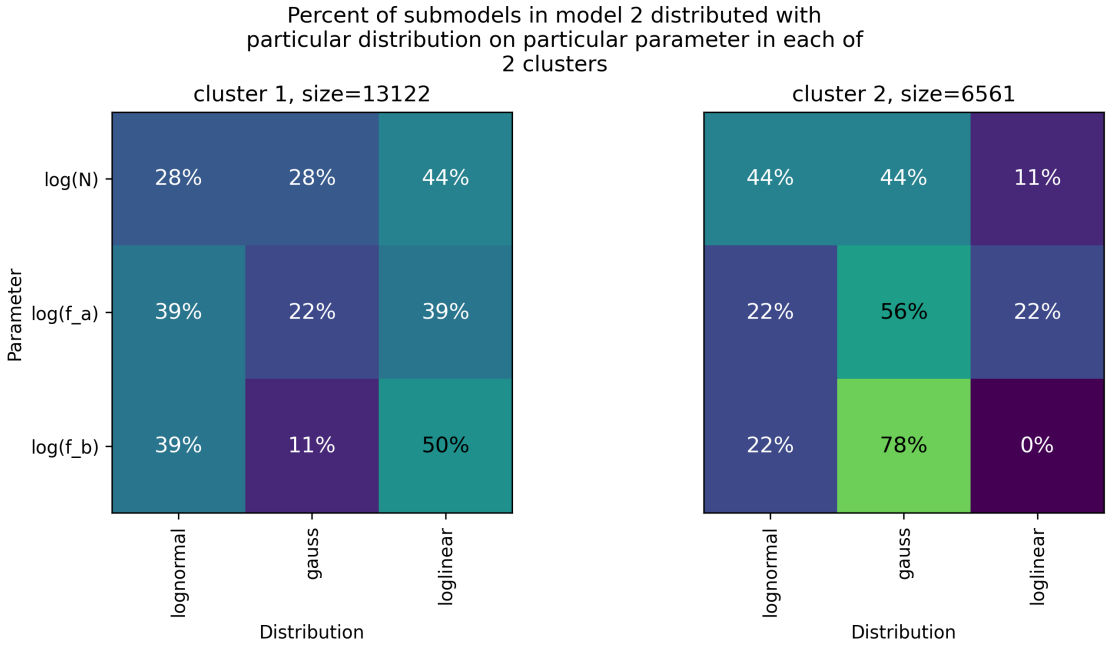

Mean density
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
1
2
3
4
5
6
7
log(L)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
log(N)

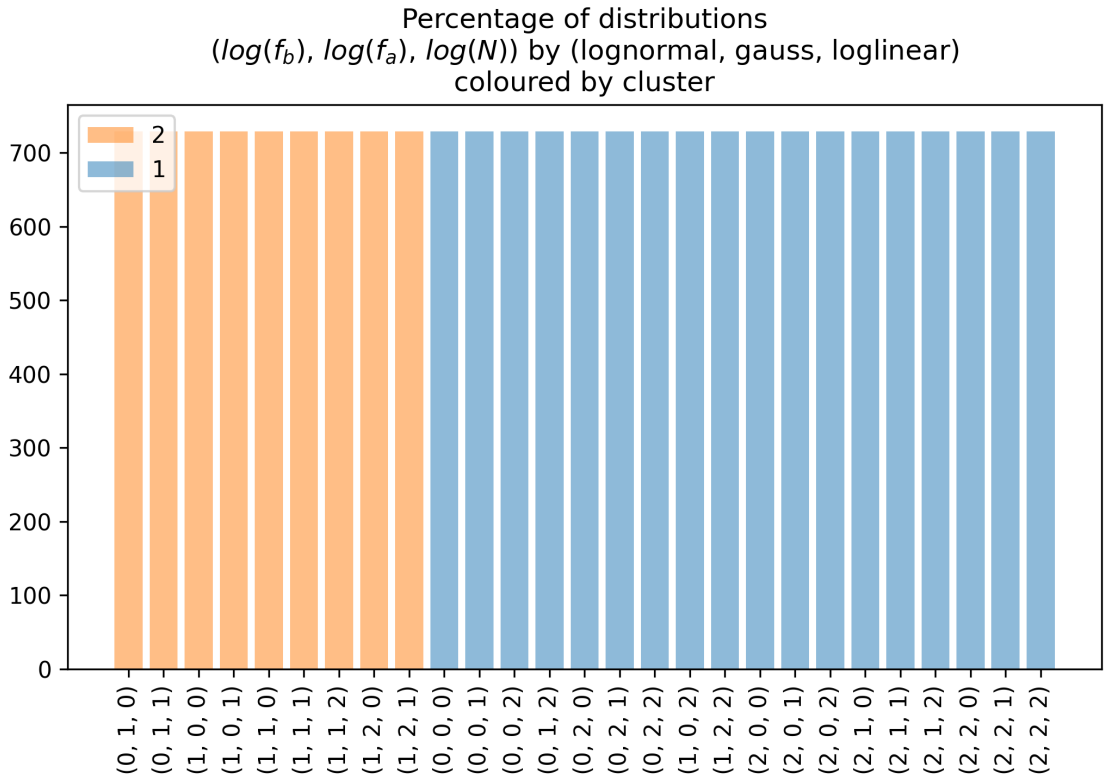

Mean density - heatmap
3.0
2.5
2.0
1.5
1.0
0.5
0.0
log(N)
0
1
2
3
4
5
6
7
log(L)


Mean density function by L
Density
log(L)
Mean survival function by L
Probability
log(L)
Mean density function by N
Density
log(N)


Means of 2 clusters
Mean histogram for each cluster
1, median = 631.0
2, median = 45.1
Mean density
log(L)
Survival function of cluster means
1
2
Probability to survive so many years
log(L)

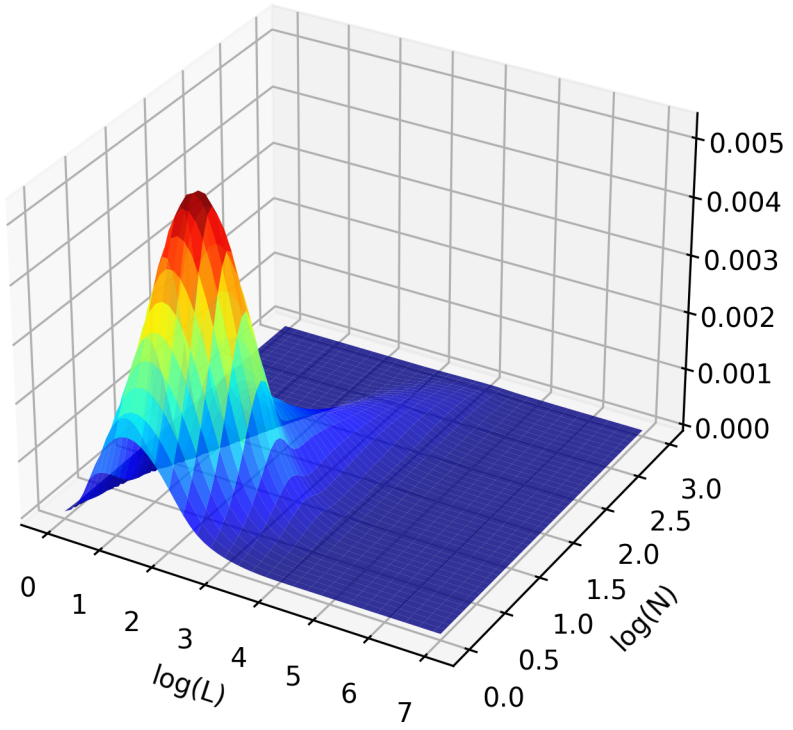

Model II after Principal component analysis (PCA)
Coloured by model
2
Coloured by cluster (2 clusters)
1
2
x1
x2
x3

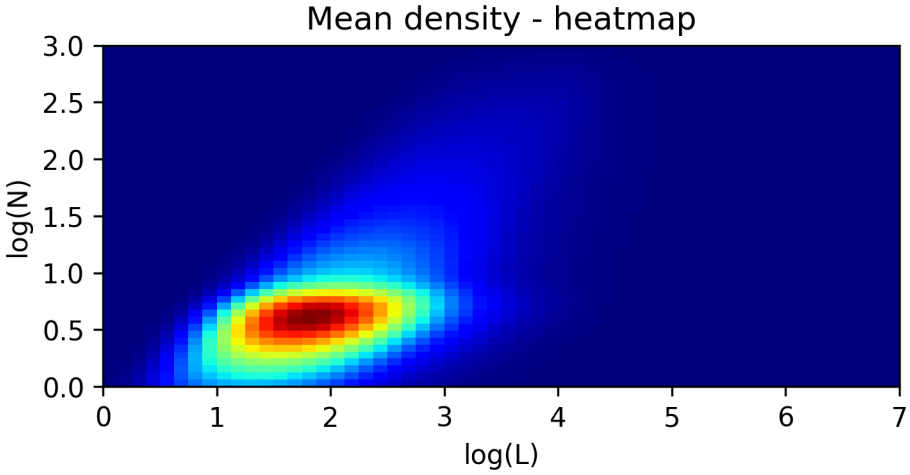

Percent of submodels in model 2 distributed with
particular distribution on particular parameter in each of
2 clusters
cluster 1, size=13122
cluster 2, size=6561
log(N)
log(f_a)
log(f_b)
Parameter
28%
28%
44%
39%
22%
39%
39%
11%
50%
44%
44%
11%
22%
56%
22%
22%
78%
0%
lognormal
gauss
loglinear
Distribution

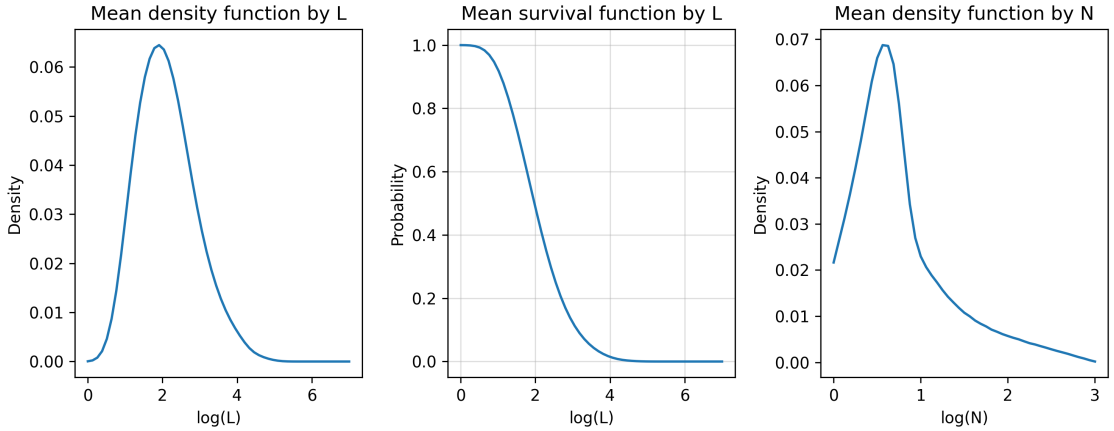


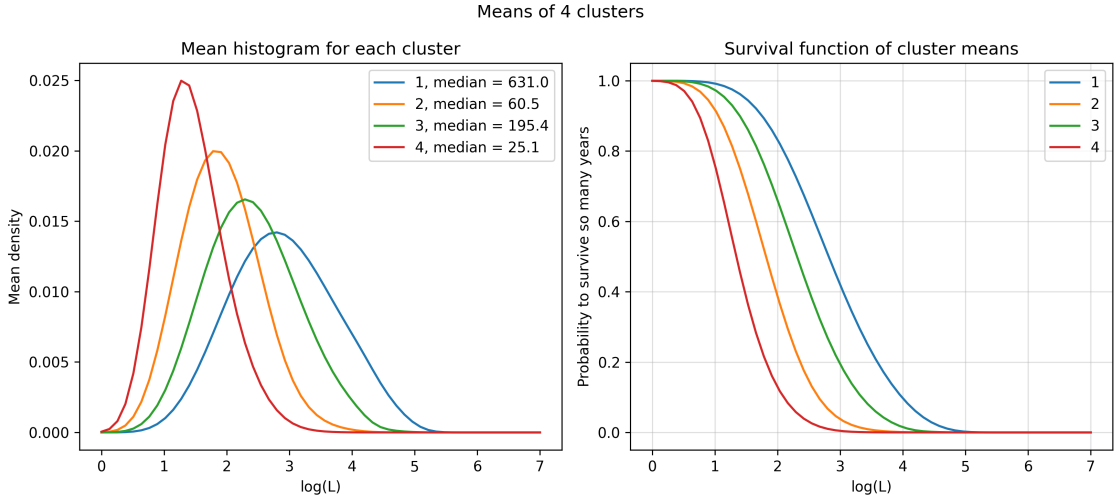
[5]:
```
for s in [1, 2]:
    cluster(model=3, ks=[4], supermodel=s)  # model 3
```

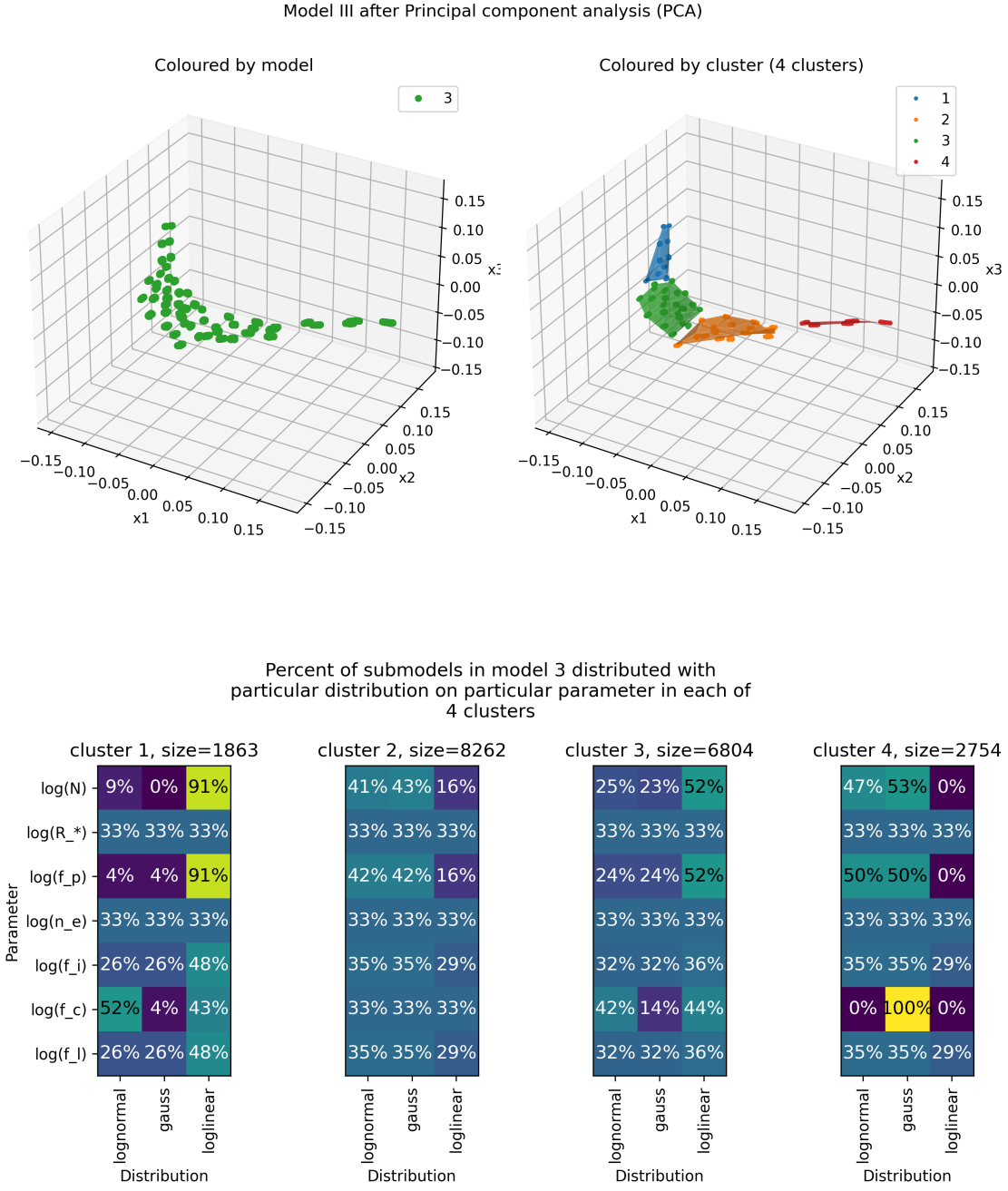
Model III

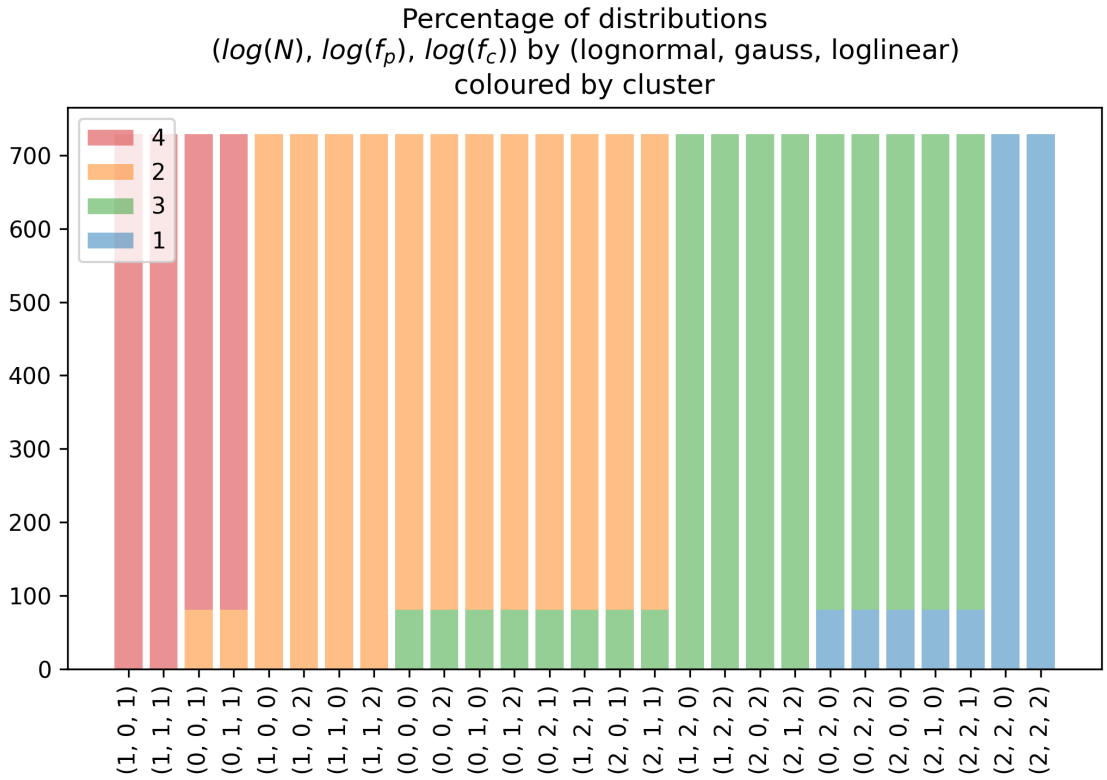

Mean density
0.005
0.004
0.003
0.002
0.001
0.000
0
1
2
3
4
5
6
7
log(L)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
log(N)


Mean density - heatmap
3.0
2.5
2.0
1.5
1.0
0.5
0.0
log(N)
0
1
2
3
4
5
6
7
log(L)


Mean density function by L
Density
log(L)
Mean survival function by L
Probability
log(L)
Mean density function by N
Density
log(N)

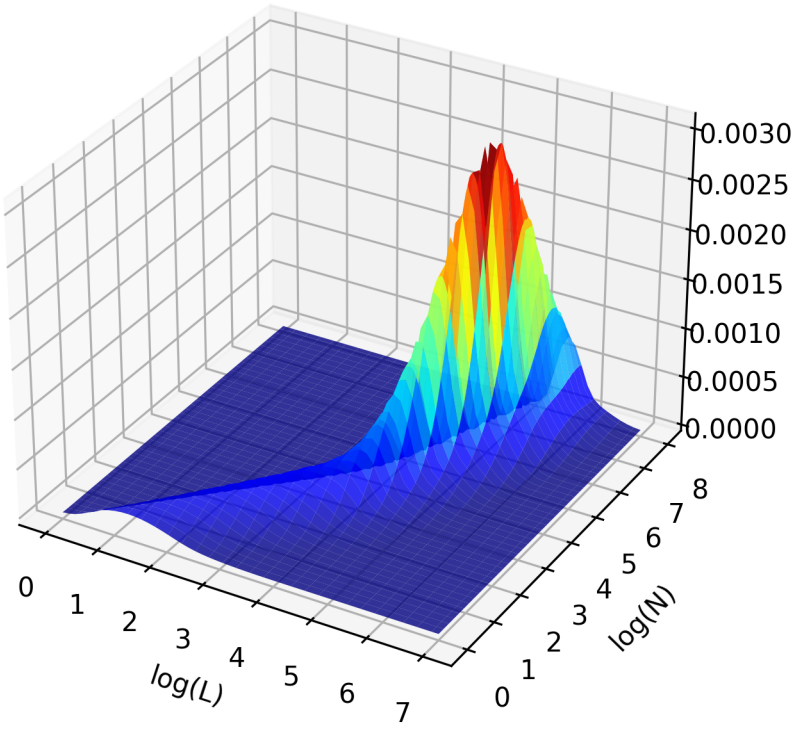

Means of 4 clusters
Mean histogram for each cluster
Mean density
log(L)
1, median = 631.0
2, median = 60.5
3, median = 195.4
4, median = 25.1
Survival function of cluster means
Probability to survive so many years
log(L)
1
2
3
4

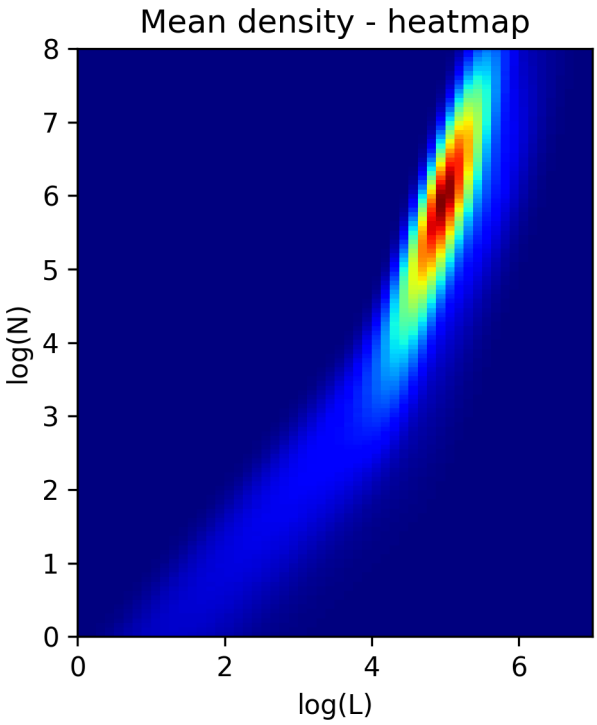

Model III after Principal component analysis (PCA)
Coloured by model
3
Coloured by cluster (4 clusters)
1
2
3
4
x1
x2
x3
Percent of submodels in model 3 distributed with particular distribution on particular parameter in each of 4 clusters
cluster 1, size=1863
cluster 2, size=8262
cluster 3, size=6804
cluster 4, size=2754
Parameter
log(N)
log(R_*)
log(f_p)
log(n_e)
log(f_i)
log(f_c)
log(f_l)
9% 0% 91%
33% 33% 33%
4% 4% 91%
33% 33% 33%
26% 26% 48%
52% 4% 43%
26% 26% 48%
41% 43% 16%
33% 33% 33%
42% 42% 16%
33% 33% 33%
35% 35% 29%
33% 33% 33%
35% 35% 29%
25% 23% 52%
33% 33% 33%
24% 24% 52%
33% 33% 33%
32% 32% 36%
42% 14% 44%
32% 32% 36%
47% 53% 0%
33% 33% 33%
50% 50% 0%
33% 33% 33%
35% 35% 29%
0% 100% 0%
35% 35% 29%
lognormal
gauss
loglinear
Distribution

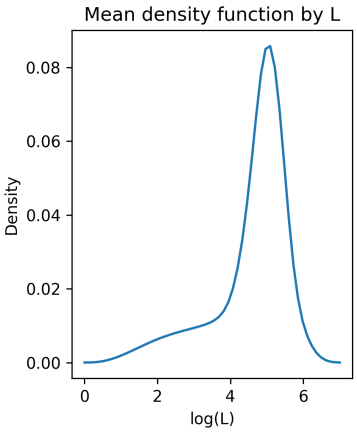

Percentage of distributions
($log(N)$, $log(f_p)$, $log(f_c)$) by (lognormal, gauss, loglinear)
coloured by cluster
4
2
3
1
0
100
200
300
400
500
600
700
(1, 0, 1)
(1, 1, 1)
(0, 0, 1)
(0, 1, 1)
(1, 0, 0)
(1, 0, 2)
(1, 1, 0)
(1, 1, 2)
(0, 0, 0)
(0, 0, 2)
(0, 1, 0)
(0, 1, 2)
(0, 2, 1)
(1, 2, 1)
(2, 0, 1)
(2, 1, 1)
(1, 2, 0)
(1, 2, 2)
(2, 0, 2)
(2, 1, 2)
(0, 2, 0)
(0, 2, 2)
(2, 0, 0)
(2, 1, 0)
(2, 2, 1)
(2, 2, 0)
(2, 2, 2)

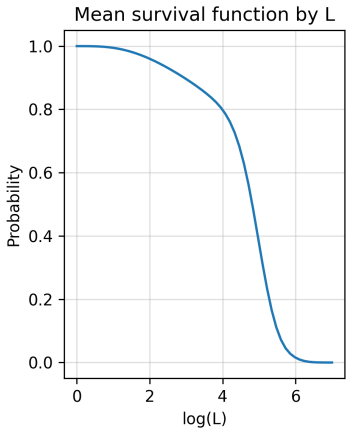
Model III

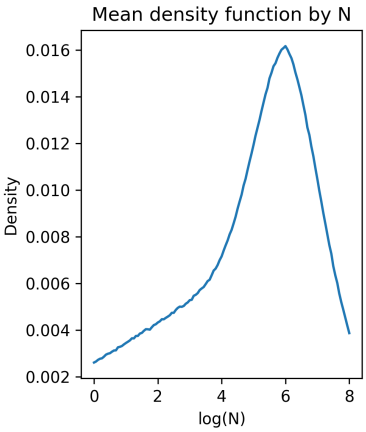
Mean density

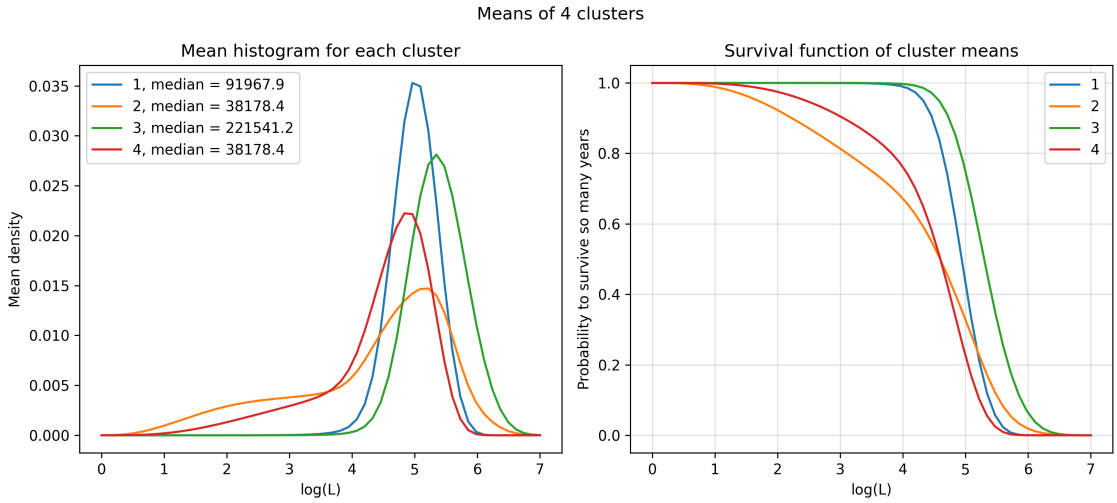


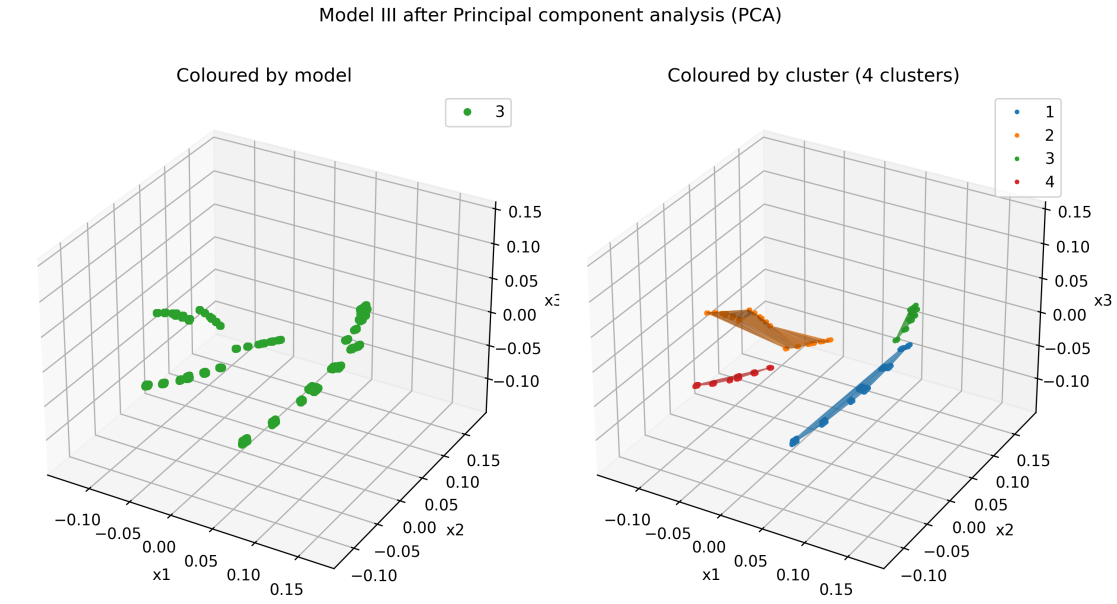

Mean density - heatmap
log(N)
log(L)
0
1
2
3
4
5
6
7
8
0
2
4
6


Mean density function by L
Density
log(L)
0.00
0.02
0.04
0.06
0.08
0
2
4
6


Mean survival function by L
Probability
log(L)
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6

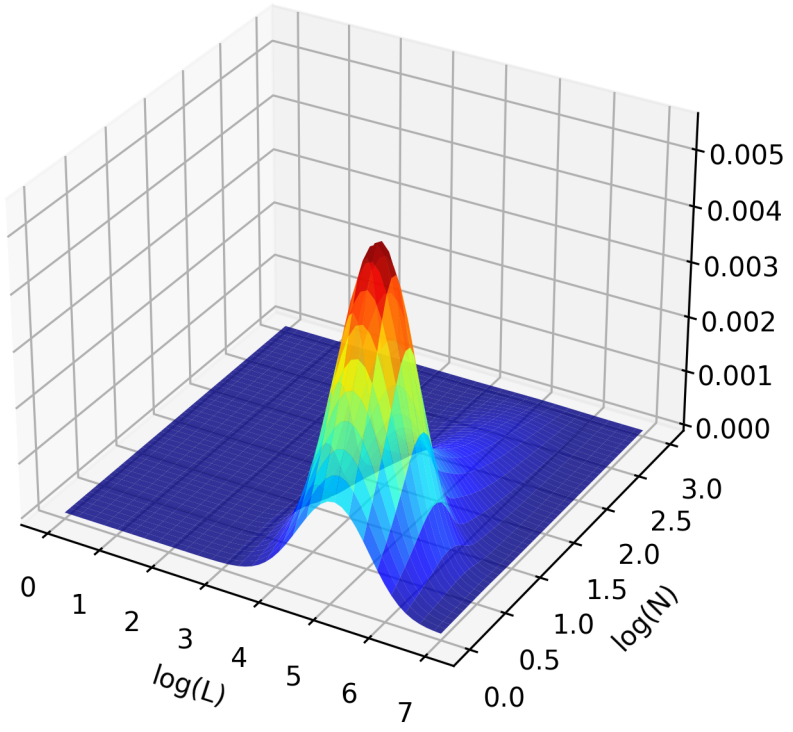

Mean density function by N
Density
log(N)
0.002
0.004
0.006
0.008
0.010
0.012
0.014
0.016
0
2
4
6
8

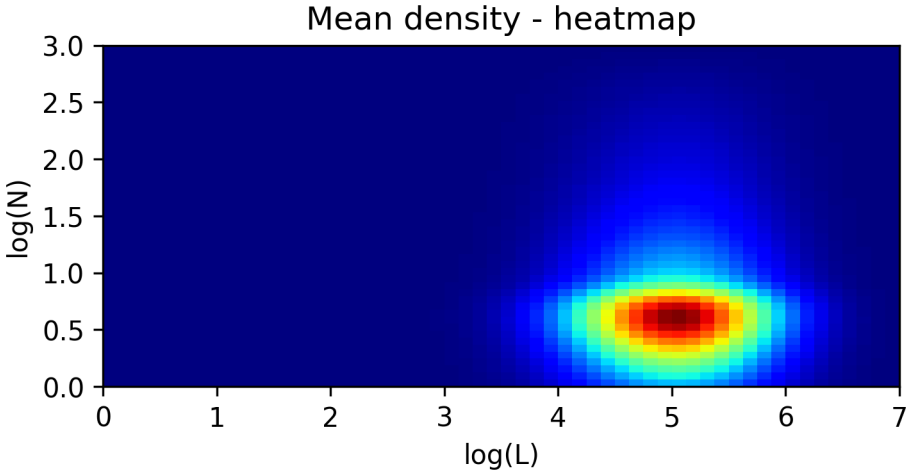


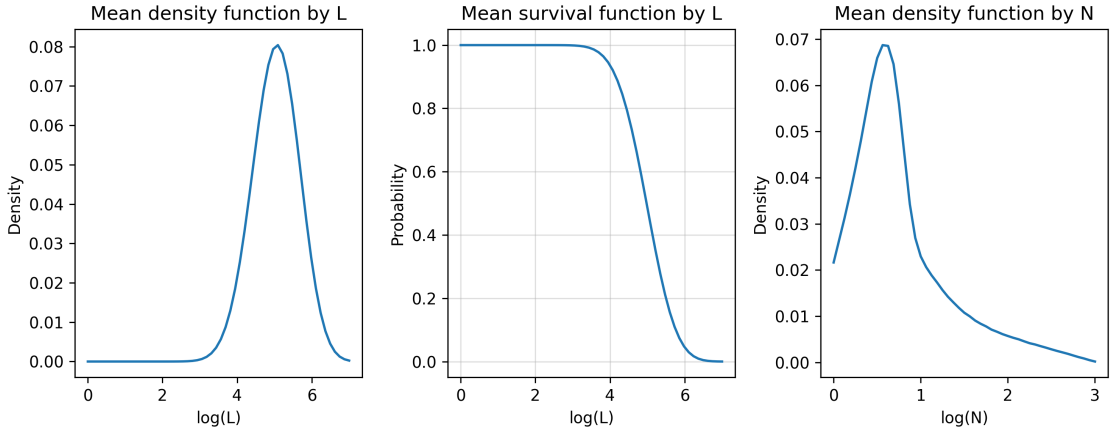


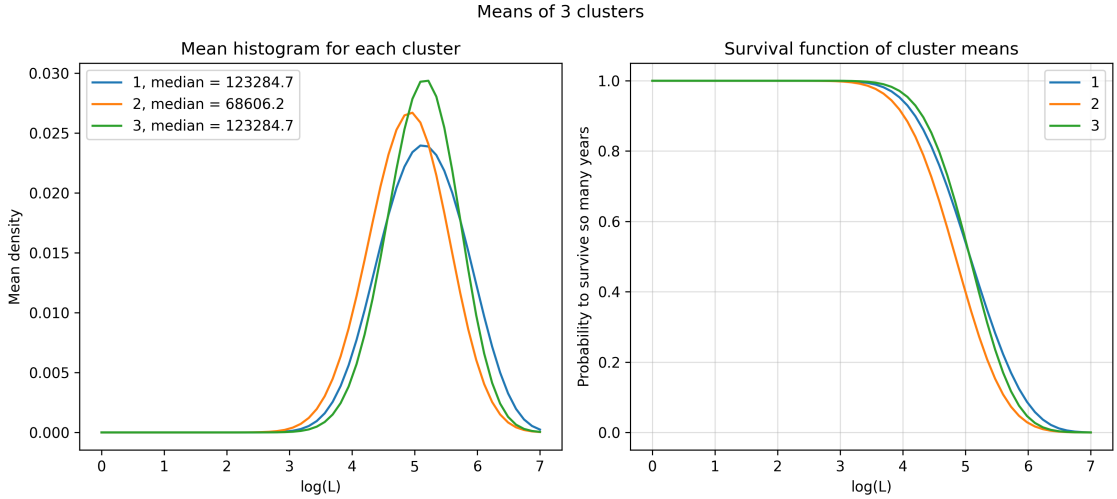
[6]:
```
cluster(model=4, ks=[3])  # model 4
```

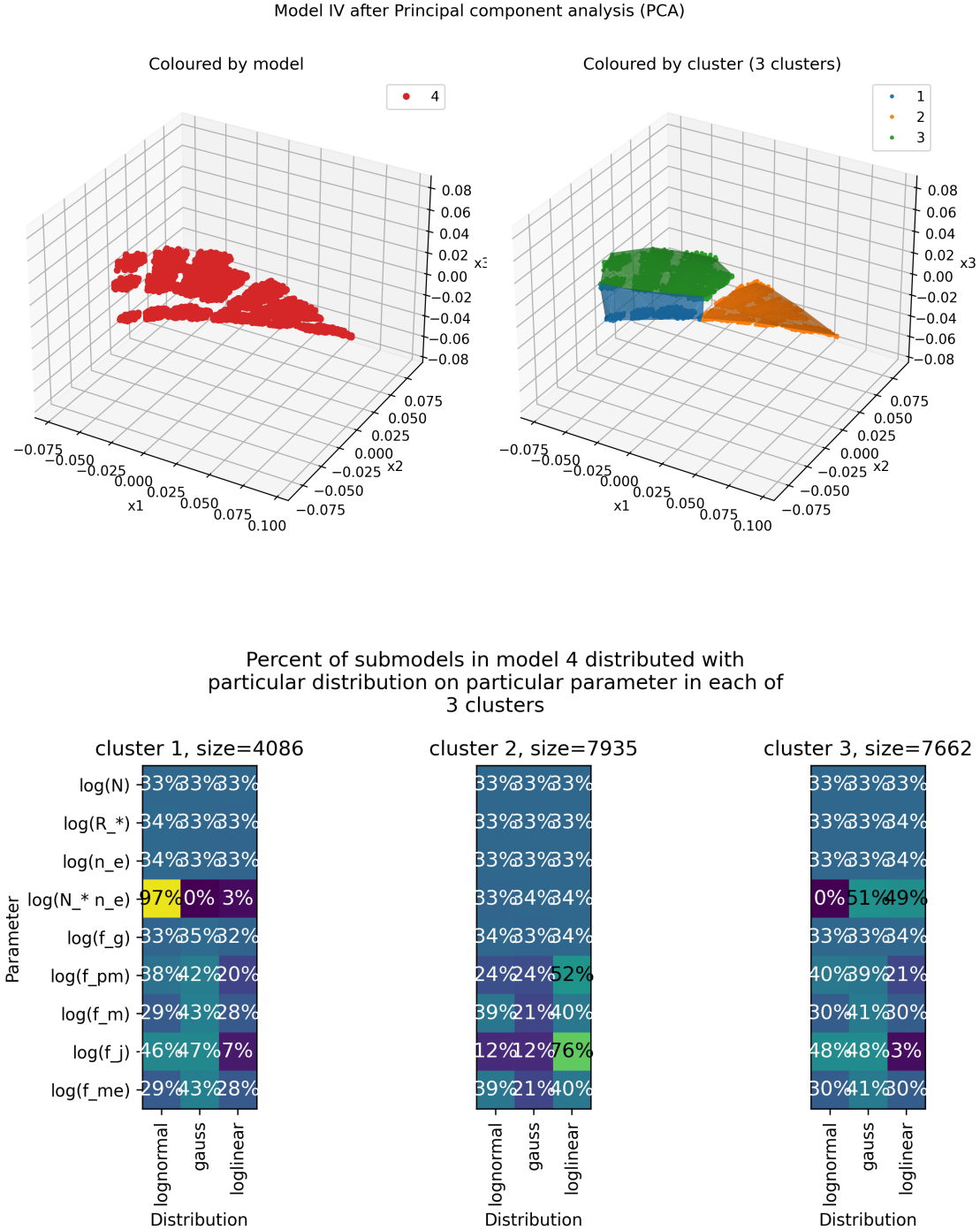
Model IV

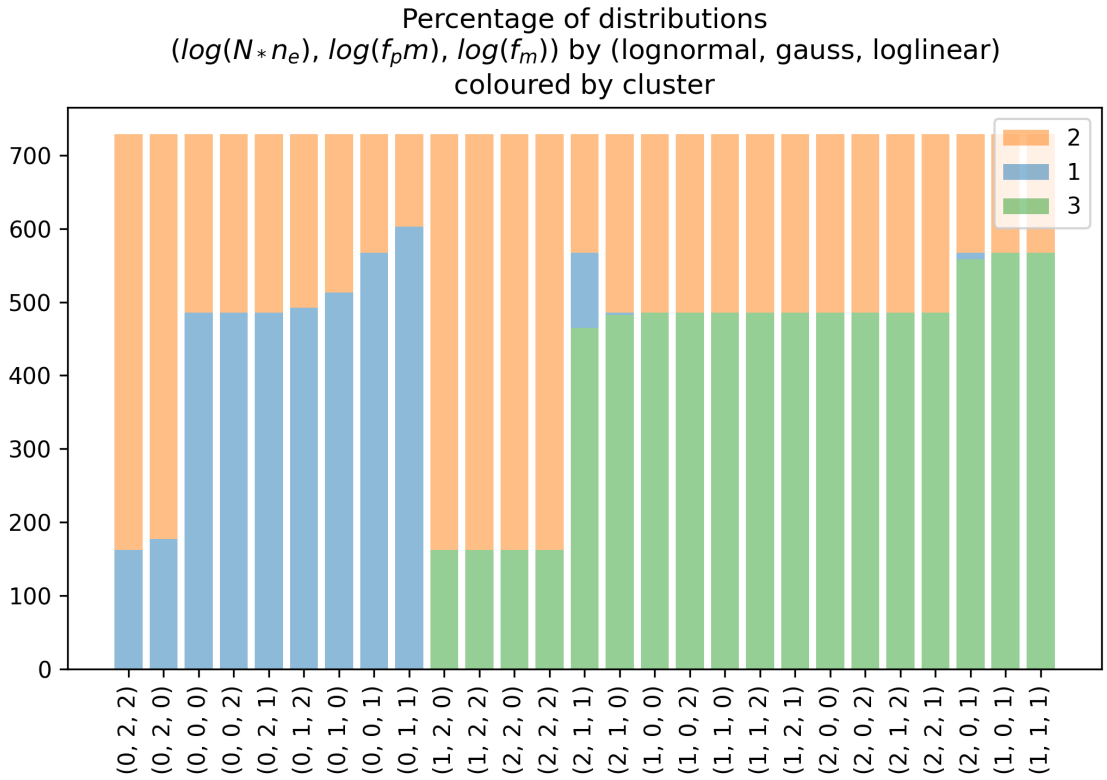

Mean density
0.005
0.004
0.003
0.002
0.001
0.000
0
1
2
3
4
5
6
7
log(L)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
log(N)


Mean density - heatmap
3.0
2.5
2.0
1.5
1.0
0.5
0.0
log(N)
0
1
2
3
4
5
6
7
log(L)


Mean density function by L
Density
log(L)
Mean survival function by L
Probability
log(L)
Mean density function by N
Density
log(N)

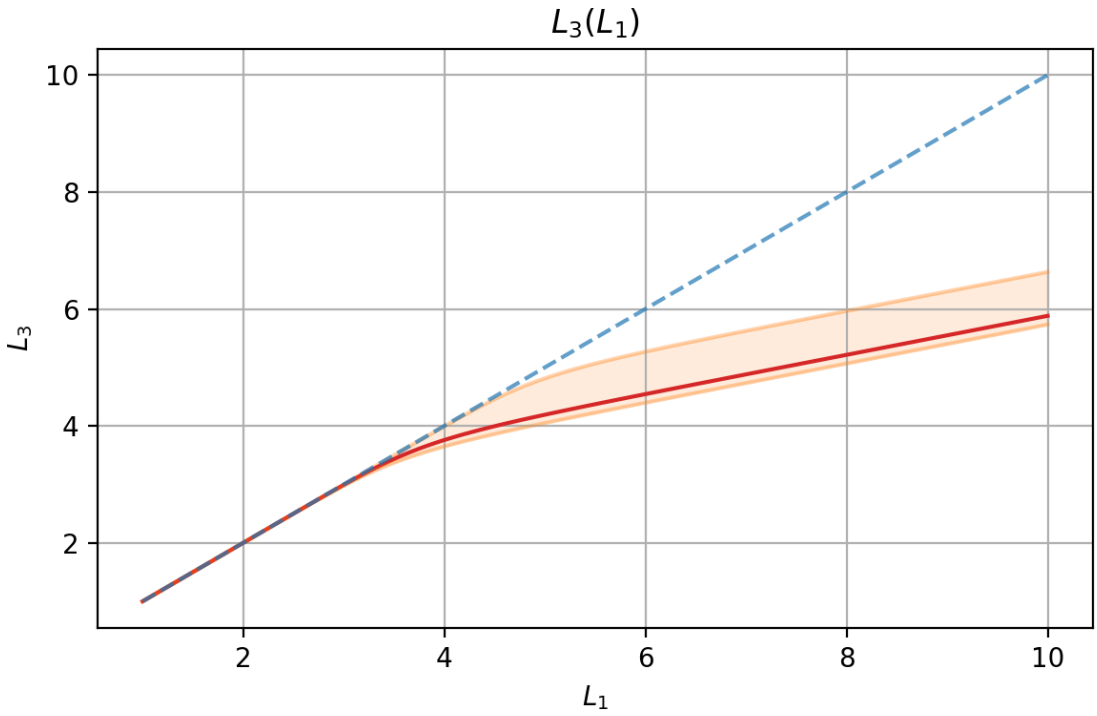

Means of 3 clusters
Mean histogram for each cluster
1, median = 123284.7
2, median = 68606.2
3, median = 123284.7
Mean density
log(L)
Survival function of cluster means
Probability to survive so many years
1
2
3
log(L)


Model IV after Principal component analysis (PCA)
Coloured by model
4
Coloured by cluster (3 clusters)
1
2
3
x1
x2
x3
Percent of submodels in model 4 distributed with particular distribution on particular parameter in each of 3 clusters
cluster 1, size=4086
cluster 2, size=7935
cluster 3, size=7662
Parameter
log(N)
log(R_*)
log(n_e)
log(N_* n_e)
log(f_g)
log(f_pm)
log(f_m)
log(f_j)
log(f_me)
lognormal
gauss
loglinear
Distribution

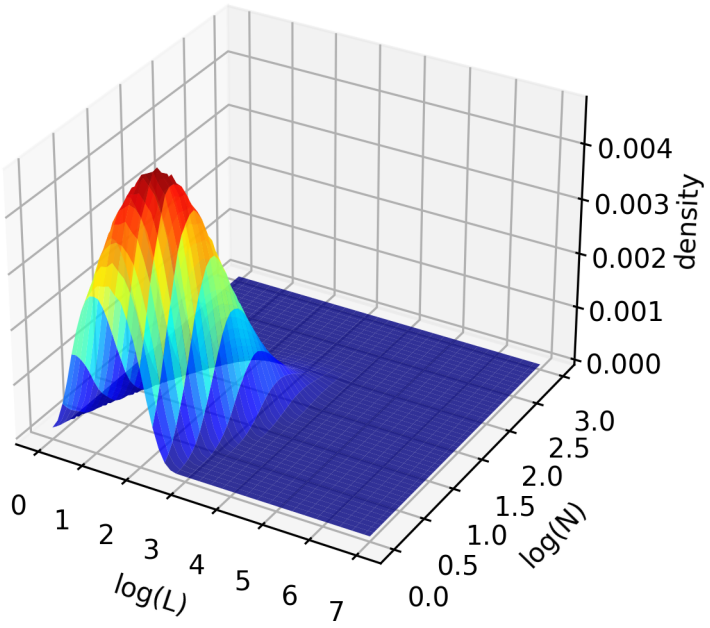


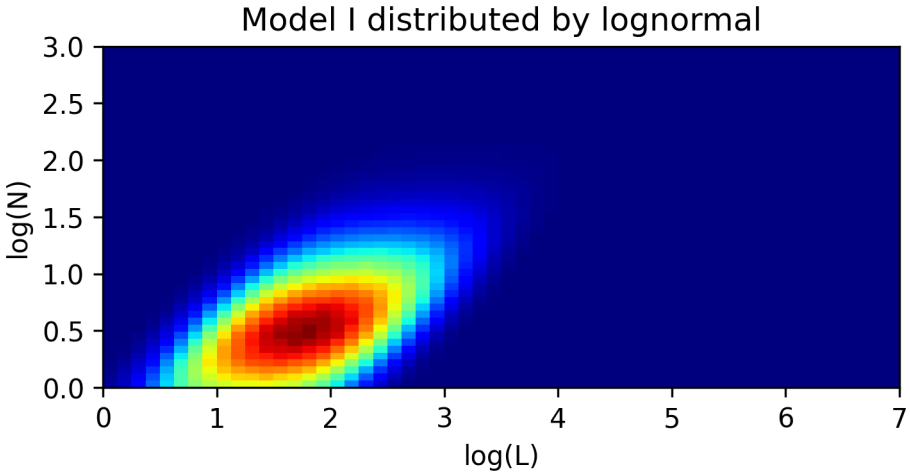
[7]:
```
def polyDeltaL(logL1):
    minP, meanP, maxP = -2, np.log10(1.8), np.log10(5)
    L1 = 10 ** logL1
    a1 = 10 ** (9 - np.array([minP, meanP, maxP]) - np.log10(np.pi * 4))
    a1 = [np.roots([1, 3 * L1, 3 * L1 ** 2 + a, L1 ** 3]) for a in a1]
    return [- min(a, key=lambda x: np.abs(x.imag)).real 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"${ylab}(L_1)$")
```

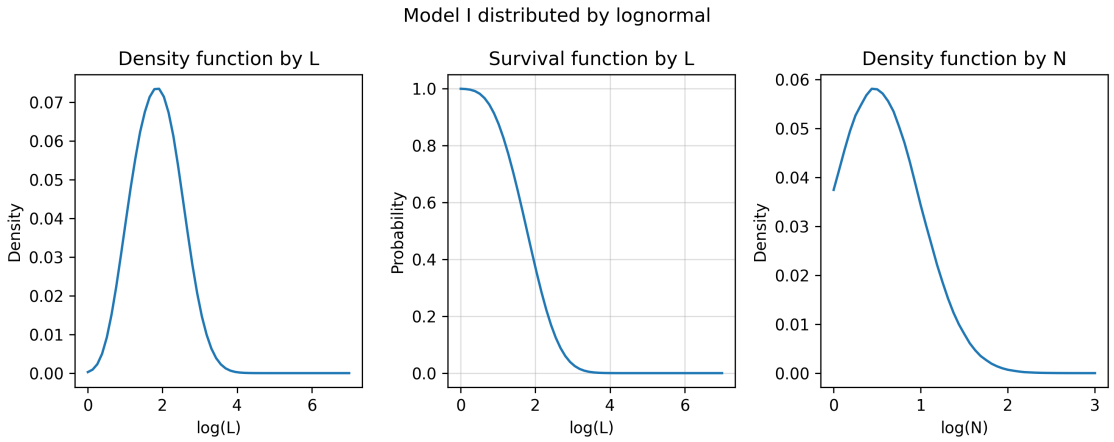
```
    plt.xlabel("$L_1$")
    plt.ylabel(f"${ylab}$")
    plt.grid()
    # plt.savefig(f"slike/model3-model1.png")
    plt.show()
```

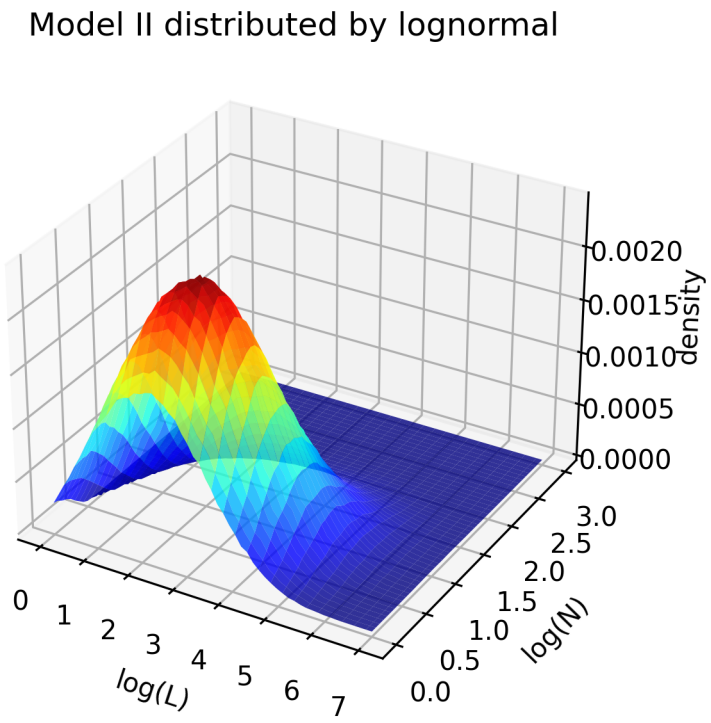


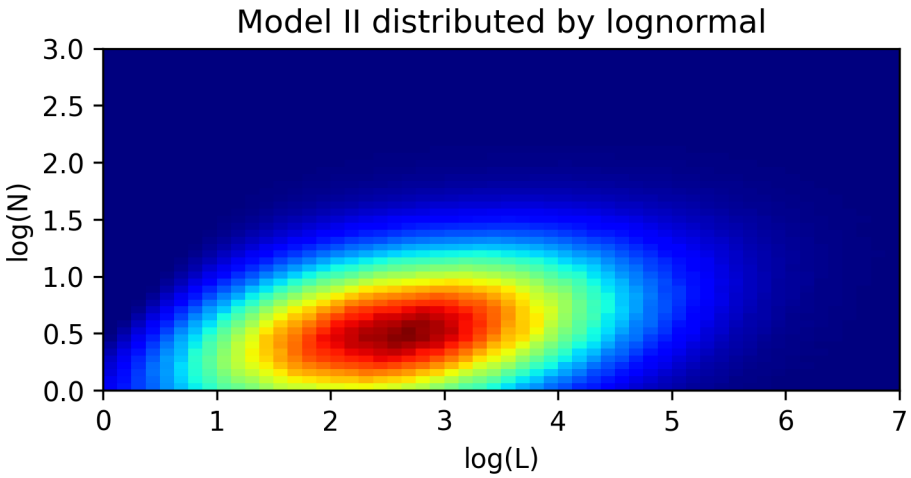
[8]:
```
from plot3D_L import *
draw_histograms3D(model=1, distribution=tuple(0 for _ in range(6)),␣
 ↪supermodel=1)
draw_histograms3D(model=2, distribution=tuple(0 for _ in range(2)),␣
 ↪supermodel=1)
draw_histograms3D(model=3, distribution=tuple(1 for _ in range(6)),␣
 ↪supermodel=1)
draw_histograms3D(model=4,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

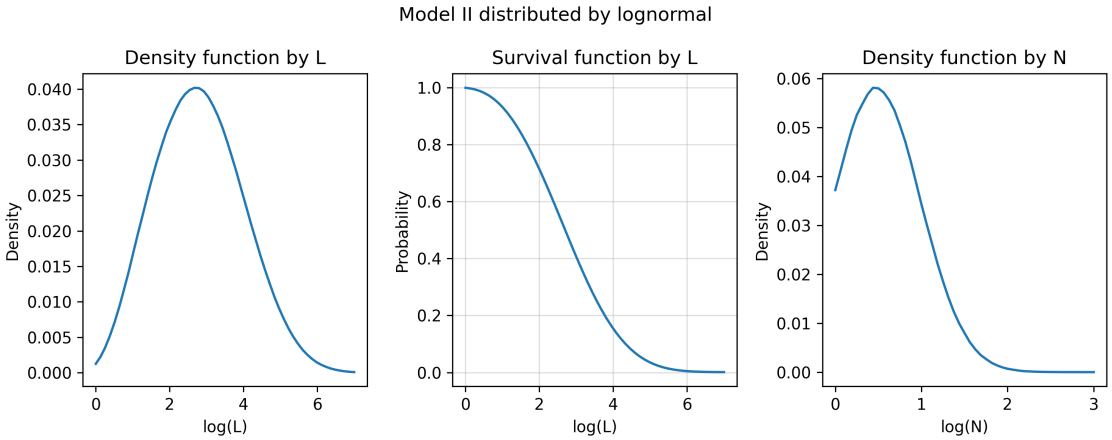

Model I distributed by lognormal
0.004
0.003
0.002
0.001
0.000
density
0
1
2
3
4
5
6
7
log(L)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
log(N)

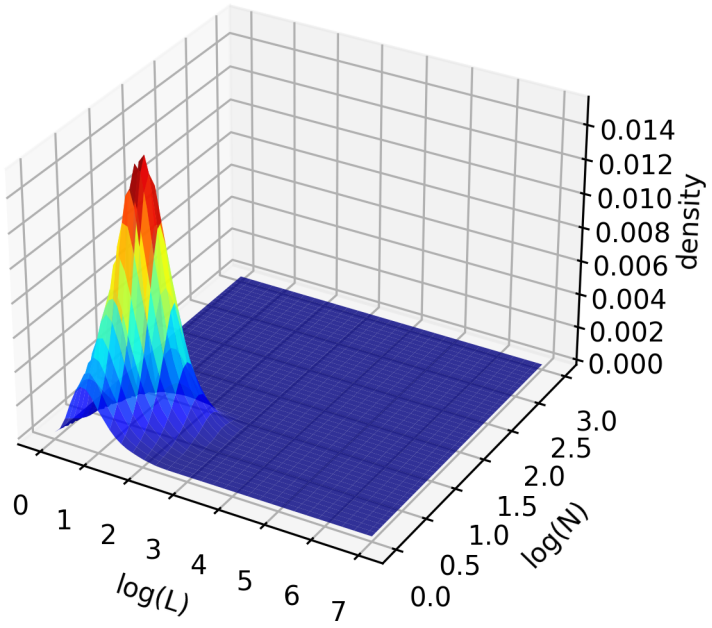

Model I distributed by lognormal
3.0
2.5
2.0
1.5
1.0
0.5
0.0
log(N)
0
1
2
3
4
5
6
7
log(L)

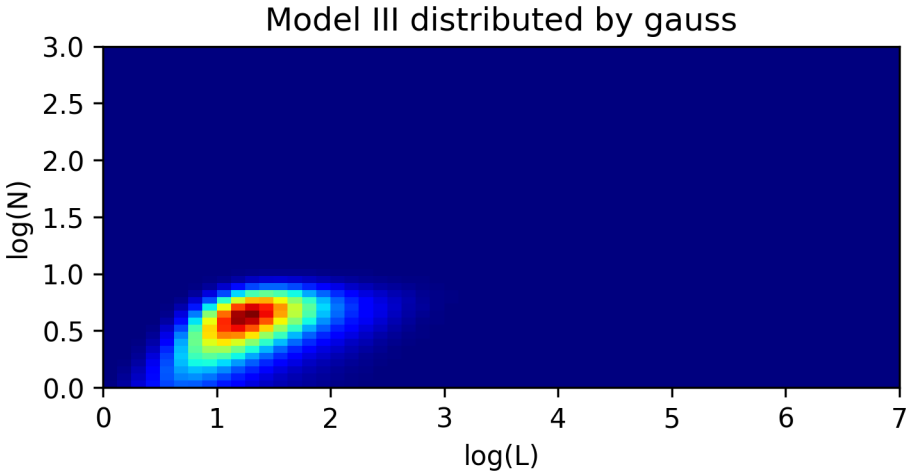

Model I distributed by lognormal
Density function by L
Survival function by L
Density function by N
Density
Probability
log(L)
log(L)
log(N)

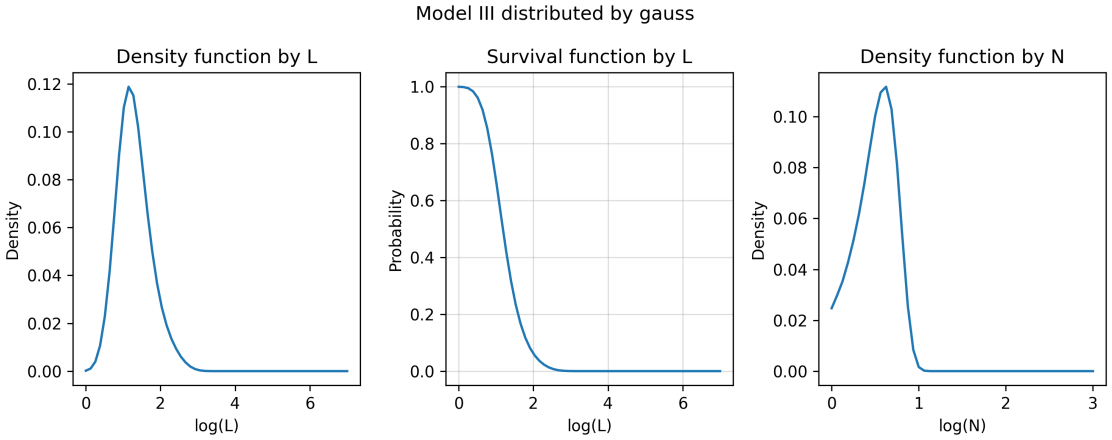

Model II distributed by lognormal
density
log(L)
log(N)

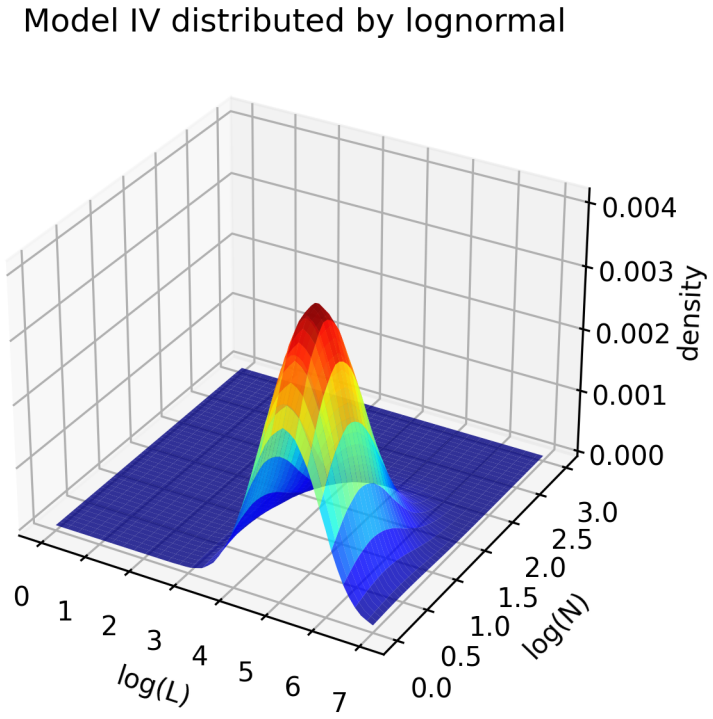

Model II distributed by lognormal
log(N)
log(L)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
5
6
7

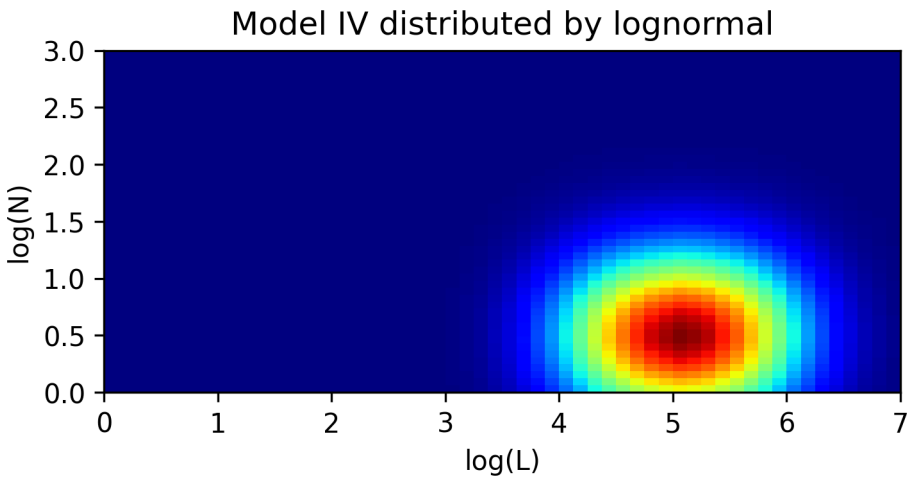

Model II distributed by lognormal
Density function by L
Density
log(L)
Survival function by L
Probability
log(L)
Density function by N
Density
log(N)

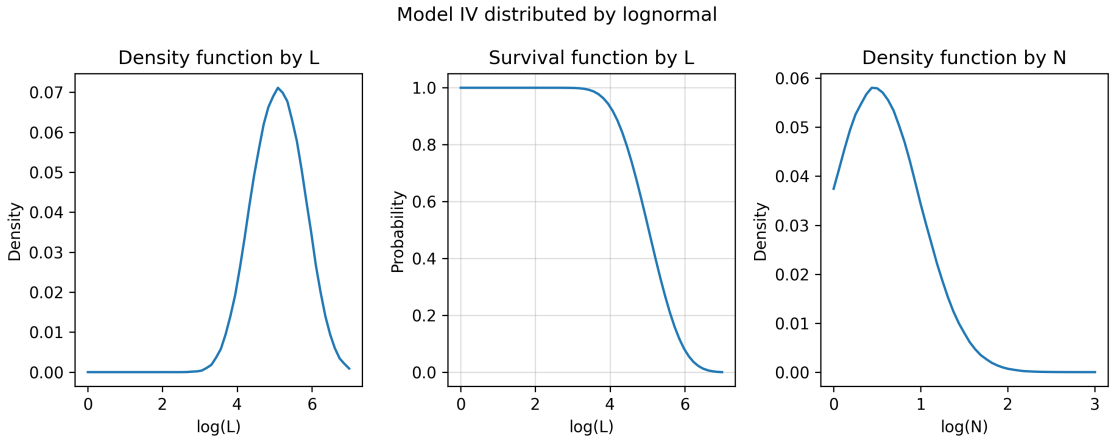

Model III distributed by gauss
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
density
0
1
2
3
4
5
6
7
log(L)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
log(N)

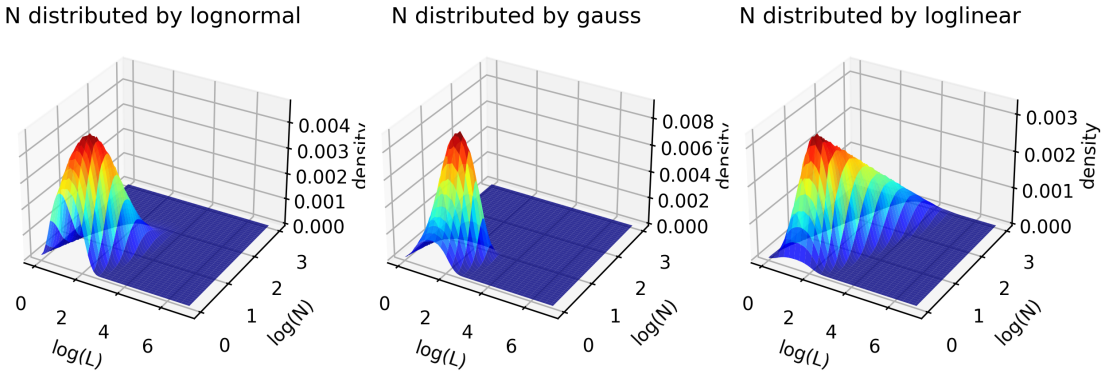

Model III distributed by gauss
3.0
2.5
2.0
1.5
1.0
0.5
0.0
log(N)
0
1
2
3
4
5
6
7
log(L)


Model III distributed by gauss
Density function by L
Survival function by L
Density function by N
Density
Probability
log(L)
log(L)
log(N)

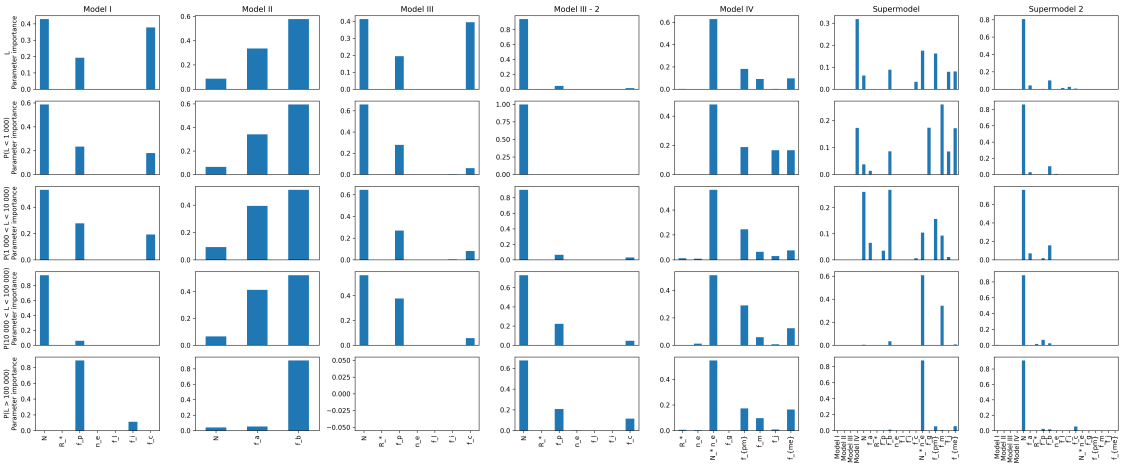

Model IV distributed by lognormal
density
log(L)
log(N)


Model IV distributed by lognormal
log(N)
log(L)


Model IV distributed by lognormal
Density function by L
Density
log(L)
Survival function by L
Probability
log(L)
Density function by N
Density
log(N)


N distributed by lognormal
N distributed by gauss
N distributed by loglinear
log(L)
log(N)
density

[9]:
```
from tabela_napake_p import modeli_napake
a = modeli_napake()
```



[10]:
```
models = []
labels = []
results = {}
with open("ML_tabele.txt", "r") as f:
    lines = f.readlines()
    for i in range(len(lines) // 13 + 1):
        part_lines = lines[i * 13:(i+1) * 13 - 1]
        model = part_lines[-2][:-1].replace("_", " ")
        label = part_lines[-1][:-1]
        if model not in models:
            models.append(model)
        if label not in labels:
            labels.append(label)
        col_names = part_lines[0][:-1].split(" ")[1:]
        row_names = [l[:-1].split(" ")[0][:13]
                     for l in part_lines[1:-2] if "error" not in l]
        part_lines = [[round(float(n), 4) for n in l[:-1].split(" ")[1:]]
                      for l in part_lines[1:-2] if "error" not in l]
        results[(model, label)] = pd.DataFrame(part_lines,
                                               index=row_names,␣
↪columns=col_names)
```

[11]:
```
"""
pd.set_option('display.width', 82)
pd.set_option('display.max_columns', None)
for model in models:
    print("\n  " + "="*60 + "\n")
    print(" "*5 + "-"*20 + " "*3 + model + " "*3 + "-"*20 + "\n")
```

```
    a = pd.concat({k[1]: l for k, l in results.items() if model in k and "<" in␣
 ↪k[1]}, 1)
    a.columns = a.columns.swaplevel()
    a.name = label
    a = a[sorted(a.columns, key=lambda x: (x[0], x[1].replace(" 0", "0")))]
    print(a)
    a.style.background_gradient(cmap="Blues")
    # from IPython.display import display
    # display(a.style.background_gradient(cmap="Oranges"))
"""
print("")
```

[12]:
```
"""
pd.set_option('display.width', 72)
pd.set_option('display.max_columns', None)

for label in labels:
    print("\n  " + "="*60 + "\n")
    print(" "*5 + "-"*20 + " "*3 + label + " "*3 + "-"*20 + "\n")
    a = pd.concat({k[0].replace("Model ", ""): l for k, l in results.items() if␣
 ↪label in k}, 1)
    a.columns = a.columns.swaplevel()
    a.name = label
    a = a[sorted(a.columns)]
    if "100 " in label:
        pd.set_option('display.width', 62)
    print(a)
    a.style.background_gradient(cmap="Blues")
    # from IPython.display import display
    # display(a.style.background_gradient(cmap="Oranges"))
"""
print("")
```