sheet07_ex14

December 19, 2023

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
```

(a)

Nehme an, dass Zählrate der Protonen Poissonverteilung folgt:
$P(X = k) = \frac{\lambda^k}{k!} \exp(-\lambda) \Rightarrow L = \prod_i P(k_i, \lambda)$
Negative-Log-Likelihood extremalisieren:
$F = -\ln(L) \Rightarrow \frac{dF}{d\lambda} = 0 \Rightarrow \hat{\lambda} = \frac{1}{n} \sum_i^n k_i$

[ ]:
```
counts = [4135, 4202, 4203, 4218, 4227, 4231, 4310]
days = [1, 2, 3, 4, 5, 6, 7]

total_counts = np.sum(counts)

lambda_L = total_counts / len(counts)

print(f"Wahrscheinlichste Zählrate: {lambda_L}")
```

```
Wahrscheinlichste Zählrate: 4218.0
```

(b)

$\lambda_i = x_i \cdot a + b \Rightarrow F(a, b) = -\sum_i^n k_i \ln(x_i a + b) - \ln(k_i!) - (x_i a + b)$

[ ]:
```
def F(x0):
    a = x0[0]
    b = x0[1]
    sum_ = 0
    k = counts
    x = days
    for i in range(len(k)):
        sum_ += -k[i]*np.log(x[i]*a+b)+(x[i]*a+b)
    return sum_

import scipy.optimize as opt
result = opt.minimize(F, x0 = [10, 4000])
#print(result)
a = result.x[0]
b = result.x[1]
```

```
print(f"Parameter a: {result.x[0]}", f"Parameter b: {result.x[1]}")
```

```
Parameter a: 21.705863006495083 Parameter b: 4131.202912494882
```

(c)

$D = 2\sum_i^n k_i \ln(ax_i + b) - (x_i a + b) - k_i \ln(\hat{\lambda_1}) + \hat{\lambda_1}$

[ ]:
```
D = 0
for i in range(len(days)):
    D += counts[i]*np.log(a*days[i]+b)-a*days[i]-b-counts[i]*np.
 ↪log(lambda_L)+lambda_L

D = 2*D
print(f"D: {D}")
print(f"sqrt(D): {np.sqrt(D)}")
```

```
D: 3.1181876552291214
sqrt(D): 1.765839079652821
```

Signifikanz $S = \sqrt{D}\sigma = 1.765839079652821\sigma$

(d)

[ ]:
```
counts = [4135, 4202, 4203, 4218, 4227, 4231, 4310, 4402]
days = [1, 2, 3, 4, 5, 6, 7, 14]

total_counts = np.sum(counts)

lambda_L = total_counts / len(counts)

print(f"Wahrscheinlichste Zählrate: {lambda_L}")
```

```
Wahrscheinlichste Zählrate: 4241.0
```

[ ]:
```
def F(x0):
    a = x0[0]
    b = x0[1]
    sum_ = 0
    k = counts
    x = days
    for i in range(len(k)):
        sum_ += -k[i]*np.log(x[i]*a+b)+(x[i]*a+b)
    return sum_

import scipy.optimize as opt
result = opt.minimize(F, x0 = [10, 4000])
#print(result)
a = result.x[0]
b = result.x[1]
```

```
print(f"Parameter a: {result.x[0]}", f"Parameter b: {result.x[1]}")
```

```
Parameter a: 19.207689368160977 Parameter b: 4140.300272806253
```

[ ]:
```
D = 0
for i in range(len(days)):
    D += counts[i]*np.log(a*days[i]+b)-a*days[i]-b-counts[i]*np.
 ↪log(lambda_L)+lambda_L

D = 2*D
print(f"D: {D}")
print(f"sqrt(D): {np.sqrt(D)}")
```

```
D: 9.975907632586313
sqrt(D): 3.158466025238567
```

Signifikanz $S = \sqrt{D}\sigma = 3.158466025238567\sigma$

(e)

Man muss hier aufpassen, da zu einer fertigen Analyse nur ein weiterer Messwert hinzugefügt wird. Da dieser Messwert der Erwartung folgt, erhöht er die Signifikanz stark, jedoch kann dieser Peak in den Messwerten auch statistische Fluktuation sein, sodass die Signifikanz fälschlich erhöht wird (Type1 Fehler). Der Hypothesentest sollte also mit weiteren Messtagen wiederholt werden, um statistische Schwankungen möglichst rauszumitteln.