UiO : **Institute of Theoretical Astrophysics**
The Faculty of Mathematics and Natural Sciences

Report for STK9021: *Mandatory Assignment*

Maksym Brilenkov

November 1, 2021

1 Introduction

In this report, I made an attempt to solve the exercises provided as part of *STK9021 – Applied Bayesian Analysis* class conducted at the University of Oslo in Autumn, 2021. The solutions comprise the combination of analytical derivations as well as `python` code for numerical computations and plots. For convenience, I tried to go point-by-point as stated in assignment problem definition, so some parts from *STK 4021-9021 Autumn 2021, Bayesian Analysis: the Oblig* document are repeated here.

If you want to reproduce my results exactly, you may need to have `python 3.8.12` installed on your system. I compiled it from source using `pyenv` tool which can be installed via:

```
$ brew install pyenv
```

if you are using MacOS, or by following their official documentation page on GitHub.

Once you installed it, you may need to modify your `.bashrc` or `.zshrc` by adding following lines:

```
export PYENV_ROOT="$HOME/.pyenv"
export PATH="$PYENV_ROOT/bin:$PATH"
export SDKROOT=/Library/Developer/CommandLineTools/SDKs/MacOSX10
   ↪ .14.sdk
export MACOSX_DEPLOYMENT_TARGET=10.14
eval "$(pyenv␣init␣--path)"
eval "$(pyenv␣virtualenv-init␣-)"
```

Then, install appropriate `python` version with:

```
$ pyenv install 3.8.12
```

Create new virtual environment:

```
$ pyenv virtualenv 3.8.12 bayes
```

And, finally, install necessary libraries with `pip` (`python` package manager):

```
$ pip install jupyter numpy scipy emcee matplotlib tqdm corner
```

The report organized as follows. In section 2, I solve exercises from Problem 1 – *The Children of Odin*. Section 3 is dedicated to Problem 2 – *A, B or C?*. Sections 4 and 5 are devoted to Problems 3 and 4 – *Bad-/good-tempered men (and their wives)* and *Rats*. I have tried to list relevant parts of the code, so it will be easier to follow what and how it was done. The entire code (`Jupyter notebooks`) is available on my GitHub[1].

[1]Project repository: *https://github.com/maksymbrl/bayesian_analysis_class*

2 The Children of Odin

Odin had six sons, but the sources are silent when it comes to daughters. So how many children did he perhaps have, in total? We assume here that if he had N children, then the number of boys y is binomial (N, p_0), with $p_0 = 0.515$, the same boy probability as in Scandinavia today.

(a) *Put up the likelihood function, for the unknown N, with the given $y = 6$ boys. What is the maximum likelihood estimate?*

In this case, the *Likelihood* is simply:

$$\mathscr{L}(N) = \binom{N}{y} p^y (1-p)^{N-y}, \quad y = 6, \tag{2.1}$$

$$\ln \mathscr{L}(N) = \ln \binom{N}{y} + y \ln p + (N-y) \ln (1-p) . \tag{2.2}$$

To get the *Maximum Likelihood Estimator* I need to minimise (2.2) with respect to N, i.e.

$$\frac{d \ln \mathscr{L}}{dN} = 0 . \tag{2.3}$$

which by analytical derivation leads to Gamma functions; thus, I solve eq.(2.3) via numerical methods in `python`.

First I import necessary libraries:

```
import numpy as np
from scipy.special import comb
from scipy.optimize import minimize
from matplotlib import pyplot as plt
```

and define some variables with the known values:

```
p = 0.515            # boy probability
y = 6                # number of sons
N = np.arange(y, 30) # number of data points
```

Once it is done, I define the $\ln \mathscr{L}$ as in eq.(2.2):

```
def lnL(N, y=6, p=0.515):
    return np.log(comb(N, y))+y*np.log(p)+(N-y)*np.log(1-p)
```

and I minimize it:

```
nglnL    = lambda N: -lnL(N, y, p)
solution = minimize(nglnL, np.array(7))
```

Note: above I used the common trick, i.e. instead of maximising likelihood, I minimise its negative value (since there are many numerical minimisation techniques, but not that many for maximisation).

Finally, the estimated actual amount of children Odin had is

```
print(solution.x)
Out[7]:  [11.13957311]
```

which is roughly double the amount of his sons. If we think about it, this makes sense, because the boy-probability is $p_0 \approx 0.5$.

This result can also be seen if we plot the $\ln \mathscr{L}(N)$ as depicted on fig.(2.1) – the value $N = 11$ corresponds to the maximum of $\ln \mathscr{L}(N)$.

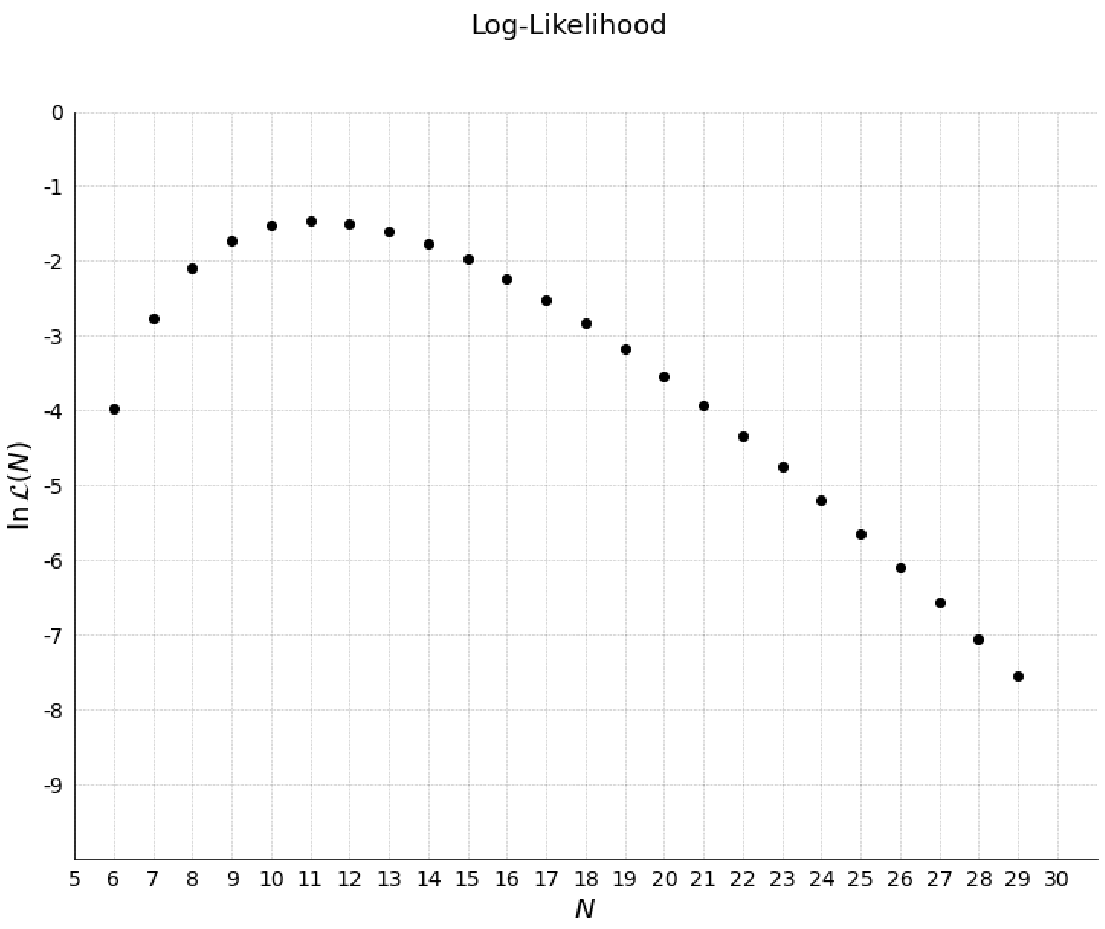

Figure 2.1: Plot of $\ln \mathscr{L}(N)$ for $N = 30$ points.

(b) *With the prior $p_0(N) \propto 1/(N+1)$, for $N = 0, 1, ..., 50$, find and portray the posterior distribution for the number of Odin's children.*

The posterior distribution can easily be calculated in python via

```
def P(N):
    return 1/(1 + N)

def L(N, y=6, p=0.515):
    return comb(N, y) * p**y * (1-p)**(N-y)

def Post(N, y=6, p=0.515):
    return L(N, y, p)*P(N)
```

and the resulting distribution is plotted in figure.2.2

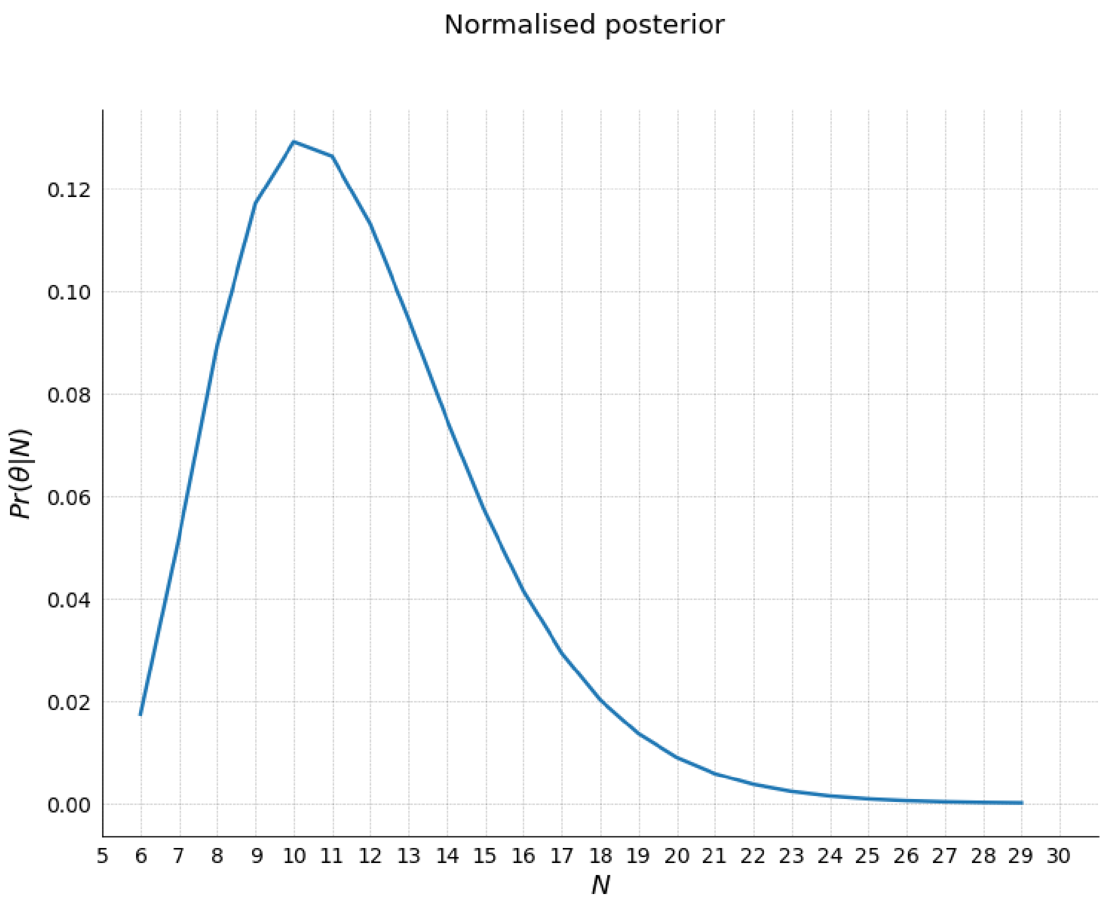

Figure 2.2: Plot of normalised posterior distribution with prior $p_0 \propto 1/(N+1)$

(c) *Think for at least three minutes in order to propose your own prior, $p_{myown}(N)$. Compute and display the resulting posterior distribution, alongside the previous one.*

After thinking for hundreds of seconds, I came up with the following assumptions regarding Odin, based on my prior beliefs and information I have obtained reading various myths/sagas about ancient times:

- Odin can have either son or daughter;
- Odin is an immortal being; thus, he can have many children throughout various epochs;
- Odin can have children not only with other gods but also with mortals (aka Greek gods);
- Odin's influence is limited only to Scandinavia since there are other gods living in different lands (e.g. Greek or Egyptian gods); thus, he cannot go there due to the political reasons;
- Odin cannot have multiple children with the same woman (in general, of course, this is not true, but here I would like to point out that this is my belief which is based on reading greek and scandinavian myths and I do not recall this condition to be untrue there);
- However, Odin cannot have children with his own daughters, grand daughters etc.;
- Odin can have more sons, than reported in sagas;

First assumption signifies some form of binomial distribution; second states that the amount of children can grow infinitely large throughout history, $N_{\text{kids}}|_{t\to\infty} \to \infty$; third and fourth states that I should consider all women but only in Scandinavia, which is a certain % of the world population; the two before last are also very crucial since they limit me to "draws without replacement" approach.

All in all, I think *Hypergeometric distribution* is reasonable prior to go with given the assumptions above. it has the form:

$$f(k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}, \tag{2.4}$$

where N is the population size, K is the number of success states in the population, n is the number of draws (i.e. quantity drawn in each trial), k is the number of observed successes.

According to internet the population in Scandinavia (Norway combined with Sweden and Denmark) as of 2020 is 21,602,034 people. I would take $\sim 0.1\%$ of this value as the total number of people in Scandinavia $\sim 1000-1500$ years ago, i.e. $N = 20,000$. So, I am interested in K of these people being children of Odin, given that I randomly choose $n = 40$ of N and $k = 6$ of them are Odin's sons (children):

$$f(K) = \frac{\binom{K}{6}\binom{20000-K}{34}}{\binom{20000}{40}}, \tag{2.5}$$

and the $\ln f(K)$ is

$$\ln f(K) = \ln\binom{K}{6} + \ln\binom{20000-K}{34} - \ln\binom{20000}{40}, \tag{2.6}$$

Python code for the prior and the posterior from the likelihood of question one is:

```
def myP(K, N, n, k):
    return comb(K, k) * comb((N-K), (n-k)) / comb(N, n)

def myPost(K, N, n, k=6, p=0.515):
    return L(K, k, p)*myP(K, N, n, k)
```

The resulting posterior distribution along with the one from previous question is plotted in figure.2.3.

(d) *Comment on the assumptions underlying the calculations here.*

I think the assumption of *Binomial* distribution is reasonable since you can get either boy or girl. The same goes to the probability $p_0 \approx 0.5$ which tells me that there are (almost) equal amount of men and women.

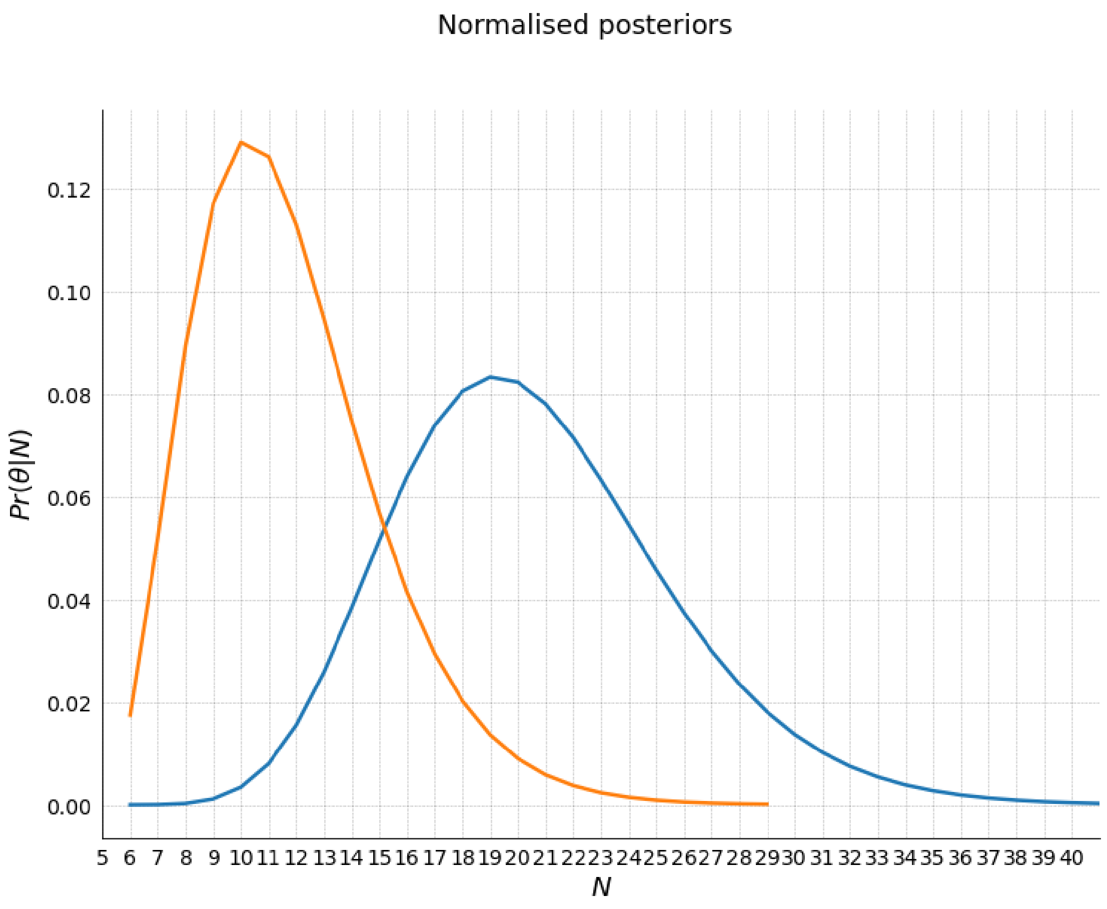

Figure 2.3: Plot of normalised posteriors' distributions with "given" and "my own" priors. The blue curve corresponds to "my own" (with prior from eq.(2.6)) whereas orange to "given" one.

3 Decision A or B or C?

Consider the square root distribution for independent non-negative observations $y_1,\ldots,y_n$, with density

$$f\left(y|\theta\right)=\frac{\theta}{2\sqrt{y}}e^{-\theta\sqrt{y}},\quad y>0\,, \tag{3.1}$$

where θ is an unknown positive parameter.

(a) *Show that $f\left(y|\theta\right)$ indeed is a density, for each given θ. Write down the likelihood function for the observed data, and show that the maximum likelihood estimator is $\hat{\theta}=1/w_n$, where $w_n=(1/n)\sum_{i=1}^n\sqrt{y_i}$.*

To check whether it is a density we need to integrate over the entire space, but because $y>0$ we will do it only from 0 to ∞:

$$\begin{aligned}\int_0^\infty f\left(y|\theta\right) &= \int_0^\infty \frac{\theta}{2\sqrt{y}}e^{-\theta\sqrt{y}}d\theta=\int_0^\infty e^{-\theta\sqrt{y}}d\left(\theta\sqrt{y}\right)\\ &= \int_0^\infty e^{-x}dx=e^{-x}|_\infty^0=1\,.\end{aligned} \tag{3.2}$$

So, it is indeed *density*. The likelihood function then:

$$\mathscr{L}(\theta)=\prod_i f\left(y_i|\theta\right)=\prod_i\frac{\theta}{2\sqrt{y_i}}e^{-\theta\sqrt{y_i}}\,, \tag{3.3}$$

$$\begin{aligned}\ln\mathscr{L}(\theta) &= \ln\left[\prod_i\frac{\theta}{2\sqrt{y_i}}e^{-\theta\sqrt{y_i}}\right]=\sum_i\left[\ln\left(\frac{\theta}{2\sqrt{y_i}}\right)+\ln\left(e^{-\theta\sqrt{y_i}}\right)\right]\\ &= \sum_i\left(\ln\theta-\ln2\sqrt{y_i}-\theta\sqrt{y_i}\right)\,,\end{aligned} \tag{3.4}$$

and

$$\frac{\partial\ln\mathscr{L}(\theta)}{\partial\theta}=\sum_i\left(\hat{\theta}^{-1}-\sqrt{y_i}\right)=0\quad\rightarrow\quad\frac{n}{\hat{\theta}}=\sum_i\sqrt{y_i},\quad\rightarrow\quad\hat{\theta}=\frac{n}{\sum_i\sqrt{y_i}}, \tag{3.5}$$

Thus,

$$\hat{\theta}=\frac{1}{w_n},\quad w_n=\frac{\sum_i\sqrt{y_i}}{n}\,. \tag{3.6}$$

(b) *Assume θ is given a Gamma prior, with parameters (a,b), i.e. with prior density*

$$p\left(\theta\right)=\frac{b^a}{\Gamma(a)}\theta^{a-1}e^{-b\theta},\quad\text{for}\quad\theta>0\,. \tag{3.7}$$

Find the posterior distribution for θ.

The posterior distribution is given by *Bayes theorem*; thus,

$$p(\theta|y_i) \propto \prod_i \frac{\theta}{2\sqrt{y_i}} e^{-\theta\sqrt{y_i}} \frac{b}{\Gamma(a)} \theta^{a-1} e^{-b\theta} = \prod_i \frac{\theta^a}{2\sqrt{y_i}} e^{-\theta(\sqrt{y_i}+b)} \frac{b}{\Gamma(a)}, \tag{3.8}$$

(c) *Assume θ has a Gamma prior with carefully set parameters* $(4.4, 2.2)$, *and that twelve costly data points have been observed from the model*:

```
0.771, 0.140, 0.135, 0.007, 0.088, 0.008, 0.268, 0.022, 0.131, 0.142,
    ↪ 0.421, 0.125
```

Display the prior and the posterior densities in a diagram. Compute also the probabilities p_1, p_2, p_3, that θ is in $(0, 1.50)$, *or* $(1.50, 3.00)$, *or* $(3.00, \infty)$, *for the prior and then for the posterior.*

First, I import libraries and add data points:

```
import numpy as np
from matplotlib import pyplot as plt
from scipy.special import gamma
import scipy.integrate as integrate

a = 4.4
b = 2.2
y = np.array([0.771, 0.140, 0.135, 0.007, 0.088, 0.008,
    ↪ 0.268, 0.022, 0.131, 0.142, 0.421, 0.125])
theta = np.linspace(0, 5, 1000)
```

Second, I define prior, posterior and likelihood distributions via eqs.(3.3),(3.7),(3.8)

```
def prob(y, theta):
    return theta/2/np.prod(y**0.5)*np.exp(-theta*np.sum(y
        ↪ **0.5))

def prior(theta,a,b):
    return b**a*theta**(a-1)*np.exp(-b*theta)/gamma(a)

def post(y, theta, a, b):
    return prob(y, theta)*prior(theta,a,b)
```

The resulting distribution is depicted in fig.(3.1).

To compute probabilities, I need to integrate over given interval, so the code will look like:

```
def int_prior(a, b, start, stop):
    return integrate.quad(lambda theta: prior(theta, a, b),
        ↪ start, stop)
```

```
def int_posterior(y, a, b, start, stop):
    # normalising th integral
    normalisation = integrate.quad(lambda theta: post(y,
        ↪ theta, a, b), 0, np.inf)
    return integrate.quad(lambda theta: post(y, theta, a, b),
        ↪  start, stop)[0]/normalisation[0]
```

In the end I get:

```
# Prior probabilities or a given interval
start=0.
stop=1.50
p1 = int_prior(a, b, start, stop)
start=1.50
stop=3.0
p2 = int_prior(a, b, start, stop)
start=3.0
stop=np.inf
p3 = int_prior(a, b, start, stop)
print(f"Prior␣is:")
print(f"-␣for␣(0,␣1.50):␣␣␣␣{p1[0]:0.4f}")
print(f"-␣for␣(1.50,␣3.00):␣{p2[0]:0.4f}")
print(f"-␣for␣(3.00,␣$\infty$):␣{p3[0]:0.4f}")
```

Output is

```
Out[79]:
Prior is:
- for (0, 1.50):    0.3399
- for (1.50, 3.00): 0.5169
- for (3.00, $\infty$): 0.1432
```

and

```
# Posterior probabilities in a given interval
start=0.
stop=1.50
p1 = int_posterior(y, a, b, start, stop)
start=1.50
stop=3.0
p2 = int_posterior(y, a, b, start, stop)
start=3.0
stop=np.inf
p3 = int_posterior(y, a, b, start, stop)
print(f"Posterior␣is:")
print(f"-␣for␣(0,␣1.50):␣␣␣␣{p1:0.4f}")
print(f"-␣for␣(1.50,␣3.00):␣{p2:0.4f}")
print(f"-␣for␣(3.00,␣$\infty$):␣{p3:0.4f}")
```

Output is

```
Out[79]:
Posterior is:
- for (0, 1.50):     0.9594
- for (1.50, 3.00): 0.0405
- for (3.00, $\infty$): 0.0000
```

The results are explained in the next question.

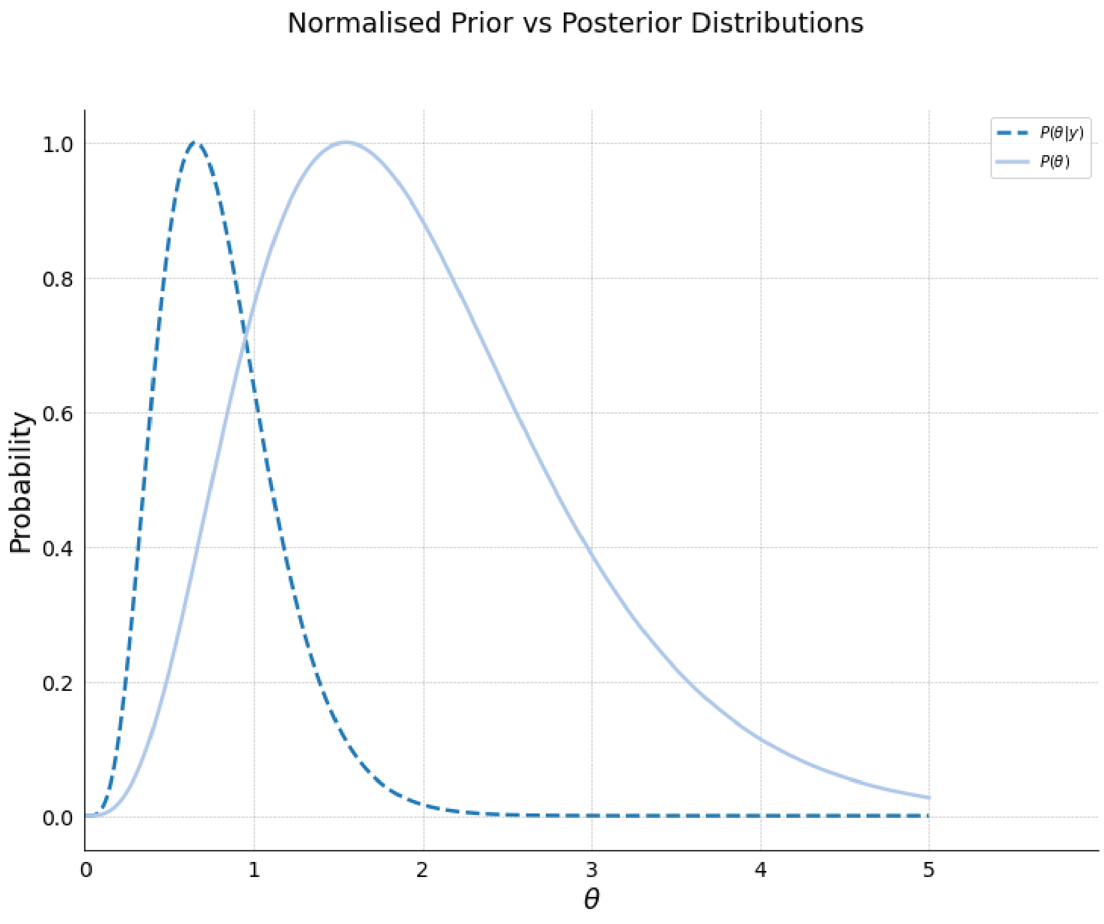

Figure 3.1: Plot of normalised Prior vs Posterior distributions.

(d) *A certain institution needs to take a decision, in January 2022, related to the size of the parameter* θ. *The three possible decisions are A, business as usual; B, investing a certain high sum in some repair; C, investing a substantially higher* *sum in a more costly operation. The loss function, associated with future costs, in annual million kroner, is*

$$L(\theta, A) = \begin{cases} 0 & \text{if} \quad \theta \leq 1.50, \\ 1 & \text{if} \quad \theta > 1.50, \end{cases}$$
$$L(\theta, B) = \begin{cases} 0 & \text{if} \quad \theta \in (1.50, 3.00), \\ 2 & \text{if} \quad \theta \notin (1.50, 3.00), \end{cases}$$
$$L(\theta, C) = \begin{cases} 0 & \text{if} \quad \theta > 3.50, \\ 3 & \text{if} \quad \theta \leq 3.50. \end{cases} \tag{3.9}$$

Which decision looked best, before the twelve data points were collected? Which decision is best, after having collected the data?

The highest probabilities from the last questions look like this:

- Prior = 0.5169 for $\theta \in (1.50, 3.00)$;
- Posterior = 0.9594 for $\theta \in (0, 1.50)$.

This means that if I had no data, I would choose event B, since loss function $C(\theta|B)$ on that interval is minimal and the probability is the highest. However, with availability of data, I would choose event A since loss $C(\theta|A)$ is minimal on that interval and the probability is the highest.

4 Bad-/good-tempered men (and their wives)

In the course of experiment, the $n = 111$ couples were interviewed in total and, as a result, the following table was produced. For the sake of simplicity I will refer to "good-tempered" persons in question simply as "good", and to "bad-tempered" as "bad" respectively:

Interviewed Couples		
M\W	Good	Bad
Good	24	27
Bad	34	26

Table 4.1: The results of the interviews conducted by Sir Francis Galton in 1887.

We can translate this table into the following matrix:

$$\begin{pmatrix} N_{0,0} & N_{0,1} \\ N_{1,0} & N_{1,1} \end{pmatrix} = \begin{pmatrix} 24 & 27 \\ 34 & 26 \end{pmatrix}, \tag{4.1}$$

with the counts $N_{i,j} = \#\{X = i, Y = j\}$ for $i, j = 0, 1$, with X is for good/bad (0/1) husbands and Y similarly for wives. We also assume the *multinomial* model with parameters $\left(n, p_{0,0}, p_{0,1}, p_{1,0}, p_{1,1}\right)$ with $p_{i,j}$ interpreted as $\Pr\left(X = i, Y = j\right)$ for a randomly selected married couple (X, Y).

(a) *Briefly discuss the validity of this assumption. Also give clear interpretations to the quantities*:

$$\begin{aligned} \alpha_i &= p_{i,0} + p_{i,1} \quad \text{for} \quad i = 0, 1 \\ \beta_j &= p_{0,j} + p_{1,j} \quad \text{for} \quad j = 0, 1 \end{aligned}, \tag{4.2}$$

The assumption above is valid for the following reasons:

- We can get in total 4 possible outcomes – it is either first, second, third or fourth couple from the table (4.1);
- The events are independent (disjoint) – if we get one couple, we won't get the other;

In this case, the quantities in eq.(4.2) simply signifies the probabilities of husbands and wives being "good" or "bad" regardless of whom they are going to marry. To be more precise:

- α_0 is the probability that husband is good regardless of the woman he marries

- α_1 is the probability that husband is bad regardless of the woman he marries
- β_0 is the probability that wife is good regardless of the man she marries
- β_1 is the probability that wife is bad regardless of the man she marries

(b) *Show that the Jeffreys prior takes the form:*

$$\pi_J\left(p_{0,0}, p_{0,1}, p_{1,0}, p_{1,1}\right) \propto \frac{1}{\sqrt{p_{0,0}p_{0,1}p_{1,0}p_{1,1}}}, \tag{4.3}$$

and that it is a Dirichlet distribution for $p = \left(p_{0,0}, p_{0,1}, p_{1,0}, p_{1,1}\right)$ *with parameters* $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$.

Indeed, in our case the *Multinomial* distribution takes the form:

$$\begin{aligned} & f\left(N_{0,0}, N_{0,1}, N_{1,0}, N_{1,1} | n, p_{0,0}, p_{0,1}, p_{1,0}, p_{1,1}\right) \\ = & \frac{n!}{N_{0,0}!N_{0,1}!N_{1,0}!N_{1,1}!} p_{0,0}^{N_{0,0}} p_{0,1}^{N_{0,1}} p_{1,0}^{N_{1,0}} p_{1,1}^{N_{1,1}}, \end{aligned} \tag{4.4}$$

where

$$\sum_i \sum_j N_{i,j} = n. \tag{4.5}$$

The Jeffreys prior is defined in terms of Fisher information:

$$\pi_{\mathrm{J}}(\theta) \propto \mathrm{I}^{1/2}(\theta), \quad \mathrm{I}(\theta) = -\mathrm{E}\left[\frac{d^2 \log p\left(X|\theta\right)}{d\theta^2}\right]. \tag{4.6}$$

Taking ln and the derivatives of (4.4) we get:

$$\begin{aligned} \ln f\left(N_{0,0}, N_{0,1}, N_{1,0}, N_{1,1} | n, p_{0,0}, p_{0,1}, p_{1,0}, p_{1,1}\right) &= \ln \frac{n!}{N_{0,0}!N_{0,1}!N_{1,0}!N_{1,1}!} \\ + \quad N_{0,0} \ln p_{0,0} + N_{0,1} \ln p_{0,1} + N_{1,0} \ln p_{1,0} + N_{1,1} \ln p_{1,1}, \end{aligned} \tag{4.7}$$

$$\frac{d \ln f}{d p_{i,j}} = \frac{N_{i,j}}{p_{i,j}}, \quad \frac{d^2 \ln f}{d p_{i,j}^2} = -\frac{N_{i,j}}{p_{i,j}^2}. \tag{4.8}$$

Recalling

$$E[X_i] = n p_i \quad \to \quad E\left[-\frac{N_{i,j}}{p_{i,j}^2}\right] = -\frac{1}{p_{i,j}^2} E\left[N_{i,j}\right] = -\frac{n}{p_{i,j}}, \tag{4.9}$$

we get

$$I = n \times \begin{pmatrix} \frac{1}{p_{0,0}} & 0 & 0 & 0 \\ 0 & \frac{1}{p_{0,0}} & 0 & 0 \\ 0 & 0 & \frac{1}{p_{0,0}} & 0 \\ 0 & 0 & 0 & \frac{1}{p_{0,0}} \end{pmatrix} \quad \to \quad \det I = \mathrm{I} = p_{0,0} p_{0,1} p_{1,0} p_{1,1}, \tag{4.10}$$

and, finally, using (4.10)

$$\pi_J \propto \sqrt{\mathrm{I}} = \frac{n}{\sqrt{p_{0,0}p_{0,1}p_{1,0}p_{1,1}}} \propto \frac{1}{\sqrt{p_{0,0}p_{0,1}p_{1,0}p_{1,1}}}. \tag{4.11}$$

Remembering that when α_i have the same values, we arrive at the special case for *Dirichlet* distribution namely *Symmetric Dirichlet Distribution*:

$$f(x_1,\ldots,x_K;\alpha) = \frac{\Gamma(\alpha K)}{\Gamma(\alpha)^K}\prod_{i=1}^{K} x_i^{\alpha-1}, \tag{4.12}$$

which in this case can be written as:

$$f\left(p_{0,0}p_{0,1}p_{1,0}p_{1,1};\frac{1}{2}\right) = \frac{\Gamma(2)}{\Gamma\left(\frac{1}{2}\right)^4}\cdot\frac{1}{\sqrt{p_{0,0}p_{0,1}p_{1,0}p_{1,1}}}, \tag{4.13}$$

Eqs. (4.11) and (4.13) are equal (up to some constant value).

(c) *We shall take an interest in the three parameters*

$$\phi = \sum_{ij}\left(p_{i,j}-\alpha_i\beta_j\right)^2/p_{i,j}, \quad \gamma = p_{0,0}+p_{1,1}, \quad \delta = \frac{p_{0,1}p_{1,0}}{p_{0,0}p_{1,1}}, \tag{4.14}$$

Explain how these parameters may be interpreted in the present context. For the Jeffreys prior, use simulation to display the 0.05, 0.50, 0.95 *quantiles of these parameters.*

- ϕ gives the correlation between parameters (probabilities);
- γ gives the probability that people with the same characters will marry each other;
- δ is the relative probability of matching with an opposite type;

To calculate the quantiles, we do the following:

```python
import numpy as np
from matplotlib import pyplot as plt
from scipy.special import gamma

np.random.seed(1234)
Nsamples = 10000
Nij = np.array([24, 27, 34, 26])
```

I will use α from Symmetric Dirichlet Distribution eq.(4.13):

```python
alpha = np.array([0.5, 0.5, 0.5, 0.5])
```

Then the code for 10000 samples will look as follows:

```
phi   = []
gamma = []
delta = []
pij   = []
for i in range(Nsamples):
    # Drawing samples from Dirichlet distribution
    pij.append(np.random.dirichlet(alpha, size=1))
    #print(pij)
    pij[i] = pij[i].flatten()
    alpha0 = pij[i][0] + pij[i][1]
    alpha1 = pij[i][2] + pij[i][3]
    beta0  = pij[i][0] + pij[i][2]
    beta1  = pij[i][1] + pij[i][3]

    phi.append((pij[i][0]-alpha0*beta0)**2/pij[i][0]+ \
               (pij[i][1]-alpha0*beta1)**2/pij[i][1]+ \
               (pij[i][2]-alpha1*beta0)**2/pij[i][2]+ \
               (pij[i][3]-alpha1*beta1)**2/pij[i][3])

    gamma.append(pij[i][0] + pij[i][3])

    delta.append((pij[i][1]*pij[i][2])/(pij[i][0]*pij[i][3]))

print(f"Phi␣␣␣(mean)␣is␣{np.mean(phi):0.2f}")
print(f"Gamma␣(mean)␣is␣{np.mean(gamma):0.2f}")
print(f"Delta␣(mean)␣is␣{np.mean(delta):0.2f}")
print(f"---------------------")
# Quantiles
print(f"Phi␣quantile:")
print(f'0.05:␣{np.quantile(phi,␣0.05):0.2f}')
print(f'0.50:␣{np.quantile(phi,␣0.50):0.2f}')
print(f'0.95:␣{np.quantile(phi,␣0.95):0.2f}')
print(f"---------------------")
print(f"Gamma␣quantile:")
print(f'0.05:␣{np.quantile(gamma,␣0.05):0.2f}')
print(f'0.50:␣{np.quantile(gamma,␣0.50):0.2f}')
print(f'0.95:␣{np.quantile(gamma,␣0.95):0.2f}')
print(f"---------------------")
print(f"Delta␣quantile:")
print(f'0.05:␣{np.quantile(delta,␣0.05):0.2f}')
print(f'0.50:␣{np.quantile(delta,␣0.50):0.2f}')
print(f'0.95:␣{np.quantile(delta,␣0.95):0.2f}')
```

and the result:

```
Out[51]:
Phi   (mean) is 6651.27
Gamma (mean) is 0.50
```

```
Delta (mean) is 7665067.43
---------------------
Phi quantile:
0.05: 0.00
0.50: 0.21
0.95: 47.55
---------------------
Gamma quantile:
0.05: 0.05
0.50: 0.50
0.95: 0.95
---------------------
Delta quantile:
0.05: 0.00
0.50: 1.00
0.95: 1480.55
```

(d) *Using Galton's data and the Jeffreys prior, derive the posterior distribution of the* $\left(p_{0,0}, p_{0,1}, p_{1,0}, p_{1,1}\right)$. *Again via simulation, display the* $0.05, 0.50, 0.95$ *quantiles for the posterior distribution of the three parameters* ϕ, γ, δ. *Sum up your findings.*

The posterior distribution is essentially the eqs.(4.4) and (4.11) multiplied together:

$$f\left(p_{0,0},\dots,p_{1,1}|n,N_{0,0},\dots,N_{1,1}\right) = n!\prod_{ij}\frac{p_{ij}^{N_{ij}}}{N_{ij}!\sqrt{p_{ij}}} = n!\prod_{ij}\frac{p_{ij}^{N_{ij}-1/2}}{N_{ij}!} \propto \prod_{ij} p_{ij}^{N_{ij}-1/2}, \tag{4.15}$$

and that looks like the Dirichlet distribution with different parameters.

We can easily reuse the code from above to calculate ϕ, γ and δ for Dirichlet distribution (4.15) if substitute $\alpha = N_{ij} + 1/2 - 1$, which will translate into `python` code as:

```
alpha = Nij + 0.5
```

As the result:

```
Phi   (mean) is 0.02
Gamma (mean) is 0.45
Delta (mean) is 1.58
---------------------
Phi quantile:
0.05: 0.00
0.50: 0.01
0.95: 0.06
---------------------
```

```
Gamma quantile:
0.05: 0.38
0.50: 0.45
0.95: 0.53
---------------------
Delta quantile:
0.05: 0.77
0.50: 1.47
0.95: 2.75
```

Since ϕ value is small, I would conclude that there is no correlation between tempers/characters and marrying. The $\gamma \approx 0.50$ within 90% confidence interval tells me that there is (almost) equally likely chance for people with the same and opposite characters to marry each other. I think that the quantiles for δ also confirms the same idea.

(e) *You're now invited to come up with your own prior for $(p_{0,0}, p_{0,1}, p_{1,0}, p_{1,1})$, which matches your prior beliefs concerning the world of married couples. For simplicity you are asked to choose your prior from the class of Dirichlet distributions, say Dir$(a_{0,0}, a_{0,1}, a_{1,0}, a_{1,1})$ with density*

$$\frac{\Gamma(k)}{\Gamma(a_{0,0})\Gamma(a_{0,1})\Gamma(a_{1,0})\Gamma(a_{1,1})} p_{0,0}^{a_{0,0}-1} p_{0,1}^{a_{0,1}-1} p_{1,0}^{a_{1,0}-1} (1 - p_{0,0} - p_{0,1} - p_{1,0})^{a_{1,1}-1}, \tag{4.16}$$

over the simplex where the $p_{i,j}$ are positive with $p_{0,0} + p_{0,1} + p_{1,0} < 1$; also, $k = a_{0,0} + a_{0,1} + a_{1,0} + a_{1,1}$. Discuss, but briefly, how you arrived at your prior.

I am going to use the distribution (4.16) with $a_{1,0} = 5$ and $a_{0,1} = 4$, while $a_{0,0} = 1/2$ and $a_{1,1} = 1/2$ because I strongly believe that couples with opposite characters marry each other more often to compensate for their shortcomings.

(f) *Please redo part of or all of the analysis using your own prior. Briefly discuss whether there are any noticeable discrepancies between the Jeffreys prior based analysis and that based on your own prior.*

Programmatically, I can reuse the same code above once again by changing my α parameter, i.e.:

```
alpha = np.array([0.5, 4, 5, 0.5])
```

which gives

```
Out[61]:
Phi   (mean) is 56830.32
Gamma (mean) is 0.10
Delta (mean) is 93783909.80
---------------------
```

```
Phi quantile:
0.05: 0.31
0.50: 5.87
0.95: 709.66
---------------------
Gamma quantile:
0.05: 0.01
0.50: 0.08
0.95: 0.29
---------------------
Delta quantile:
0.05: 10.64
0.50: 467.14
0.95: 329946.45
```

The posterior distribution in this case will be (from eqs. (4.4) and (4.16)):

$$f\left(p_{0,0},\ldots,p_{1,1}|n,N_{0,0},\ldots,N_{1,1}\right) \propto \prod_{ij} p_{ij}^{N_{ij}+a_{ij}-1}, \tag{4.17}$$

Once again, reusing the code above with different α:

```
alpha = np.array([0.5, 4, 5, 0.5]) + Nij
```

which gives:

```
Out[62]:
Phi   (mean) is 0.03
Gamma (mean) is 0.42
Delta (mean) is 2.02
---------------------
Phi quantile:
0.05: 0.00
0.50: 0.02
0.95: 0.10
---------------------
Gamma quantile:
0.05: 0.35
0.50: 0.42
0.95: 0.50
---------------------
Delta quantile:
0.05: 1.03
0.50: 1.88
0.95: 3.51
```

5 Rats

The data to be analysed are as follows. At each of ten dosage levels $x_1, \dots, x_{10}$, equal to

```
0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0
```

$m = 6$ rats are exposed to the compound at that level, and the number of these six rats that experience Event A is

```
2, 1, 2, 2, 3, 4, 1, 2, 5, 3
```

respectively. We hence have ten binomial experiments, say

$$y_j \propto \text{binom}\left(m, p_j\right), \quad \text{for} \quad j = 1, \dots, 10, \tag{5.1}$$

and these are modelled as

$$p_j = \Pr\left(\text{A}|x_j\right) = \text{H}\left(a + bx_j\right) = \frac{e^{a+bx_j}}{1 + e^{a+bx_j}}, \quad j = 1, \dots, 10, \tag{5.2}$$

with $H(u) = \frac{e^u}{(1+e^u)}$ the logistic transform.

(a) *Show that the log-likelihood function is*

$$\ln \mathscr{L}(a,b) = \sum_{j=1}^{10} \left[y_i \log p_j(a,b) + \left(m - y_j\right) \log\left\{1 - p_j(a,b)\right\} + \log \binom{m}{y_j} \right], \tag{5.3}$$

Programme this log-likelihood function and find its maximisers.

We start with some definitions. *Binomial Distribution* is defined as

$$P\left(y_j; p_j, m\right) = \binom{m}{y_j} \cdot p^{y_j} \cdot \left(1 - p\right)^{m - y_j}, \tag{5.4}$$

The *Likelihood* and *Log-Likelihood* are defined as follows:

$$\mathscr{L}(\theta) = \prod_{j=1}^{m} p_\theta\left(y_j\right), \tag{5.5}$$

$$\log \mathscr{L}(\theta) = \sum_{j=1}^{m} \log p_\theta\left(y_j\right), \tag{5.6}$$

So,

$$\begin{aligned}
\log \mathscr{L}(\theta) &= \sum_{j=1}^{m} \log\left[\binom{m}{y_j} \cdot p^{y_j} \cdot \left(1 - p\right)^{m - y_j}\right] \\
&= \sum_{j=1}^{m} \left[\log \binom{m}{y_j} + \log p^{y_j} + \log\left(1 - p\right)^{m - y_j}\right] \\
&= \sum_{j=1}^{m} \left[\log \binom{m}{y_j} + y_j \log p + \left(m - y_j\right) \log\left(1 - p\right)\right], \qquad (5.7)
\end{aligned}$$

Once again we solve it numerically. First, as usual, importing libraries and data:

```
import emcee
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import comb
from scipy.optimize import minimize
from IPython.display import display, Math

np.random.seed(123)

m=6
x = np.array([0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8,
    ↪ 2.0])
y = np.array([2, 1, 2, 2, 3, 4, 1, 2, 5, 3])
```

and solving eq.(5.7)

```
def lnL(theta, x, y):
    """
    Log-Likelihood of two variables: a and b
    """
    a, b = theta[0], theta[1]
    # Probability modelled as logistic transform
    p_j = lambda a, b, x: np.exp(a + b * x) / (1 + np.exp(a +
        ↪  b * x))
    # Putting up Log-Likelihood
    return np.sum(y * np.log(p_j(a, b, x)) + (m-y) * np.log
        ↪ (1-p_j(a, b, x)) + np.log(comb(m, y)))

# Creating the initial guess
initial_guess = np.array([1, 1])
# Packing two variables into one: theta = theta(a,b)
nglnL = lambda theta: -lnL(theta, x, y)
# Using the trick -- minimising the negative value
solution = minimize(nglnL, initial_guess)
# Printing the results
print(f"The␣Maximum␣Likelihood␣estimators␣are:")
print(f"-␣a␣value␣is:␣{solution.x[0]:0.3f}")
print(f"-␣b␣value␣is:␣{solution.x[1]:0.3f}")
```

gives:

```
Out[2]:
The Maximum Likelihood estimators are:
- a value is: -1.173
- b value is: 0.747
```

the same result as the assignment states.

(b) *With a flat prior for (a, b) on $[-8, 8] \times [-8, 8]$, set up a Markov Chain Monte Carlo scheme to assess the posterior distribution of (a, b). Record the posterior means and posterior standard deviations, for the two model parameters, and compare to values obtained by the 'Lazy Bayesian' strategy, that of normal approximations from maximum likelihood theory.*

To do MCMC I am going to use `emcee python` library. As a starting point I will use $\hat{a}$ and $\hat{b}$ values obtained from the last question.

```
pos = np.array([a_hat, b_hat]) + 1e-4 * np.random.randn(32,
    ↪ 2)
```

Therefore, by writing the prior and posterior probabilities given the constraints above:

```
def lnPrior(theta, x):
    """
    The logistic transform
    """
    a, b = theta
    #p_j = lambda a, b, x: np.exp(a + b * x) / (1 + np.exp(a
       ↪ + b * x))
    if -8.0 <= a <= 8.0 and -8.0 <= b <= 8.0:
        return 0.0 # The flat prior means some constant value
            ↪  => probability is exactly the same as
            ↪ likelihood
    return -np.inf # does not exist outside the box of [-8,
       ↪ 8] x [-8, 8]

def lnPosterior(theta, x, y):
    # Getting prior
    lp = lnPrior(theta, x)
    if not np.isfinite(lp):
        return -np.inf
    # The product becomes sum in log space
    return lp + lnL(theta, x, y)
```

The MCMC with $N_{\text{samples}} = 100000$ becomes trivial

```
nwalkers, ndim = pos.shape
sampler = emcee.EnsembleSampler(
    nwalkers, ndim, lnPosterior, args=(x, y)
)
sampler.run_mcmc(pos, Nsamples, progress=True)
```

As a result, I can calculate the mean and standard deviation for a and b:

```
a_mcmc_mean = flatchain.mean(axis=0)[0]
b_mcmc_mean = flatchain.mean(axis=0)[1]
```

```
a_mcmc_std   = flatchain.std(axis=0)[0]
b_mcmc_std   = flatchain.std(axis=0)[1]
print(f"---------------------")
print(f"MCMC␣Mean:")
print(f"--␣a:␣{a_mcmc_mean:0.3f}")
print(f"--␣b:␣{b_mcmc_mean:0.3f}")
print(f"---------------------")
print(f"MCMC␣Std:")
print(f"--␣a:␣{a_mcmc_std:0.3f}")
print(f"--␣b:␣{b_mcmc_std:0.3f}")
```

Output is

```
Out[67]:
---------------------
MCMC Mean:
-- a: -1.228
-- b: 0.786
---------------------
MCMC Std:
-- a: 0.619
-- b: 0.487
```

The "Lazy Bayesian" strategy, as was explained in "Bayesian Data Analysis" book[2], is based on the idea that for large enough n the maximum likelihood estimate gives all relevant information about θ available from the data, i.e. in repeated sampling $\theta = \theta_0$, the sampling distribution $\hat{\theta}(y)$ is approximately normal with mean θ_0 and precision $nJ(\theta_0)$.

To compare it to "Lazy Bayesian" strategy:

```
a_lazy = []
b_lazy = []
for i in range(Nsamples):
    lazy_bayes = np.random.multivariate_normal(mean=optimiser
       ↪ .x, cov=optimiser.hess_inv)
    a_lazy.append(lazy_bayes[0])
    b_lazy.append(lazy_bayes[1])

print(f"---------------------")
print(f"Lazy␣Mean:")
print(f"--␣a:␣{np.mean(a_lazy,␣axis=0):0.3f}")
print(f"--␣b:␣{np.mean(b_lazy,␣axis=0):0.3f}")
print(f"---------------------")
print(f"Lazy␣Std:")
print(f"--␣a:␣{np.std(a_lazy,␣axis=0):0.3f}")
print(f"--␣b:␣{np.std(b_lazy,␣axis=0):0.3f}")
```

[2] http://www.stat.columbia.edu/ gelman/book/

Output is

```
Out[65]:
---------------------
Lazy Mean:
-- a: -1.172
-- b: 0.747
---------------------
Lazy Std:
-- a: 0.607
-- b: 0.477
```

which looks pretty close to the ones obtained from MCMC and Maximum Likelihood estimation in first question.

(c) *Use your simulations to produce a* 90% *pointwise credibility band around the estimated curve* $\hat{p}(x)$, *with low*(x) *the* 0.05 *quantile and up*(x) *the* 0.95 *quantile of the posterior distribution for* $H(a+bx)$.

The result of the simulations along with specified credibility intervals is depicted in figure 5.1.

(d) *We also take an interest in* $p(x) = Pr(A|x)$ *for higher dosage levels than for the range* 0.2 *to* 2.0. *Give the posterior distribution of* $p(x_{new})$ *for the high dosage level* $x_{new} = 2.50$, *in terms of a histogram or estimated density. Discuss briefly the assumptions underlying your analysis.*

Since I simulate $N = 100000$, from the underlying idea of the "Lazy Bayesian" method, I conclude the distribution to be normal. Therefore, I assume Gaussian distribution and the code is simply:

```
p_new = []

for i in range(Nsamples):
    a_new = np.random.normal(a_mcmc_mean,a_mcmc_std)
    b_new = np.random.normal(b_mcmc_mean,a_mcmc_std)

    p_new.append(expit(a_new + b_new * 2.5))
```

The resulting histogram is plotted in figure 5.2.

(e) *In addition to finding the posterior distribution for* $p(x_{new})$ *above, find the predictive distribution for* y_{new}, *the number of* $m = 6$ *Ratti norvegici experiencing Event A when the dosage is* $x_{new} = 2.50$.

The predictive distribution is Binomial; thus, the predictive distribution will have the form depicted in figure 5.3

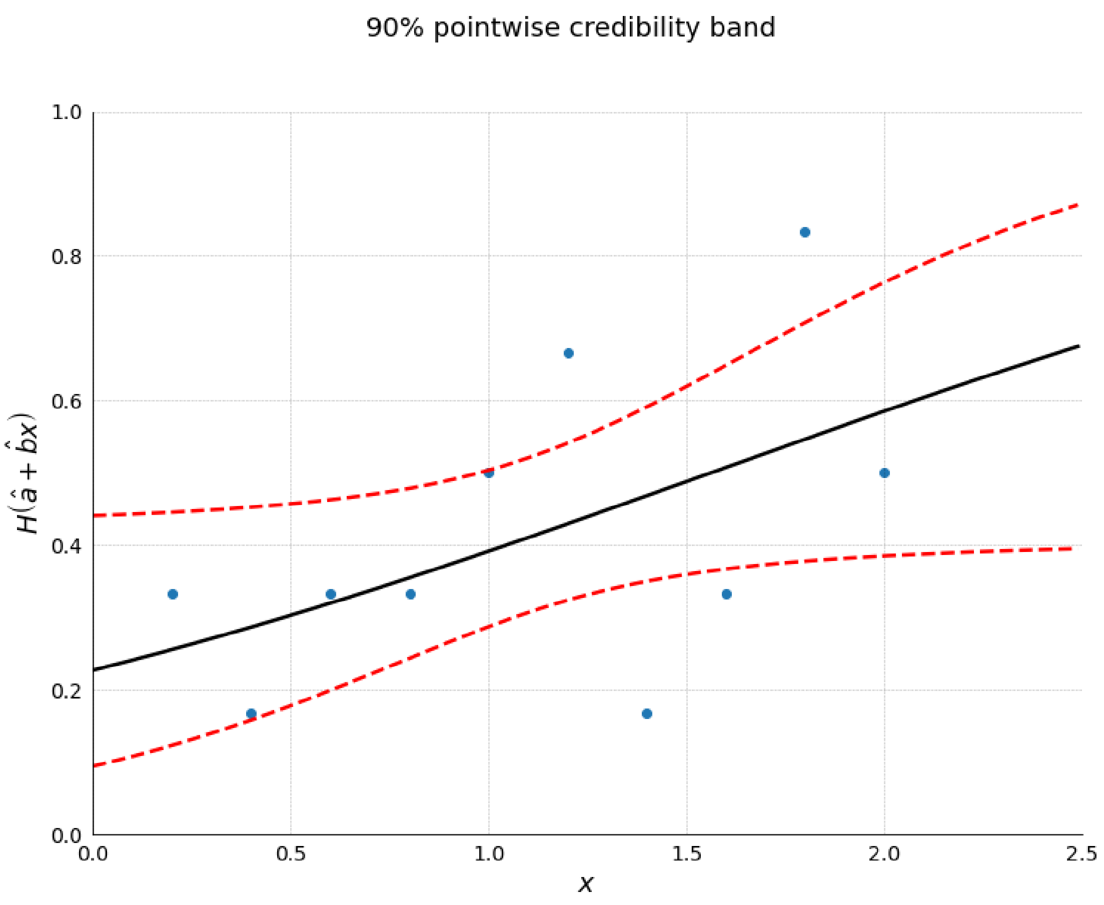

Figure 5.1: 90% pointwise credibility band around the estimated curve $\hat{p}(x) = H\left(\hat{a}+\hat{b}x\right)$, for MCMC sampled $\hat{a}$ and $\hat{b}$ with $N = 100000$ samples.

(f) *Above your Bayesian computations have been carried out with a flat prior for* (a, b) *on* $[-8, 8] \times [-8, 8]$. *In this particular context it is natural to assume that b cannot be negative, however, as more poison should increase the probability for Event A. Set up a second MCMC to compute the posterior distribution for* (a, b) *when the prior is flat on* $[-8, 8] \times [0, 8]$. *With this prior, compute by simulation the* 0.05, 0.50, 0.95 *quantile points of the posterior distribution for the point so-called LD50 parameter, or 'lethal dose 50-percent', the dosage level* x_0 *where* 50% *of the objects are expected to experience Event A.*

The MCMC code is almost the same as in question (a), so i am going to reuse it, with slight change, i.e. change

```
if -8.0 <= a <= 8.0 and -8.0 <= b <= 8.0:
```

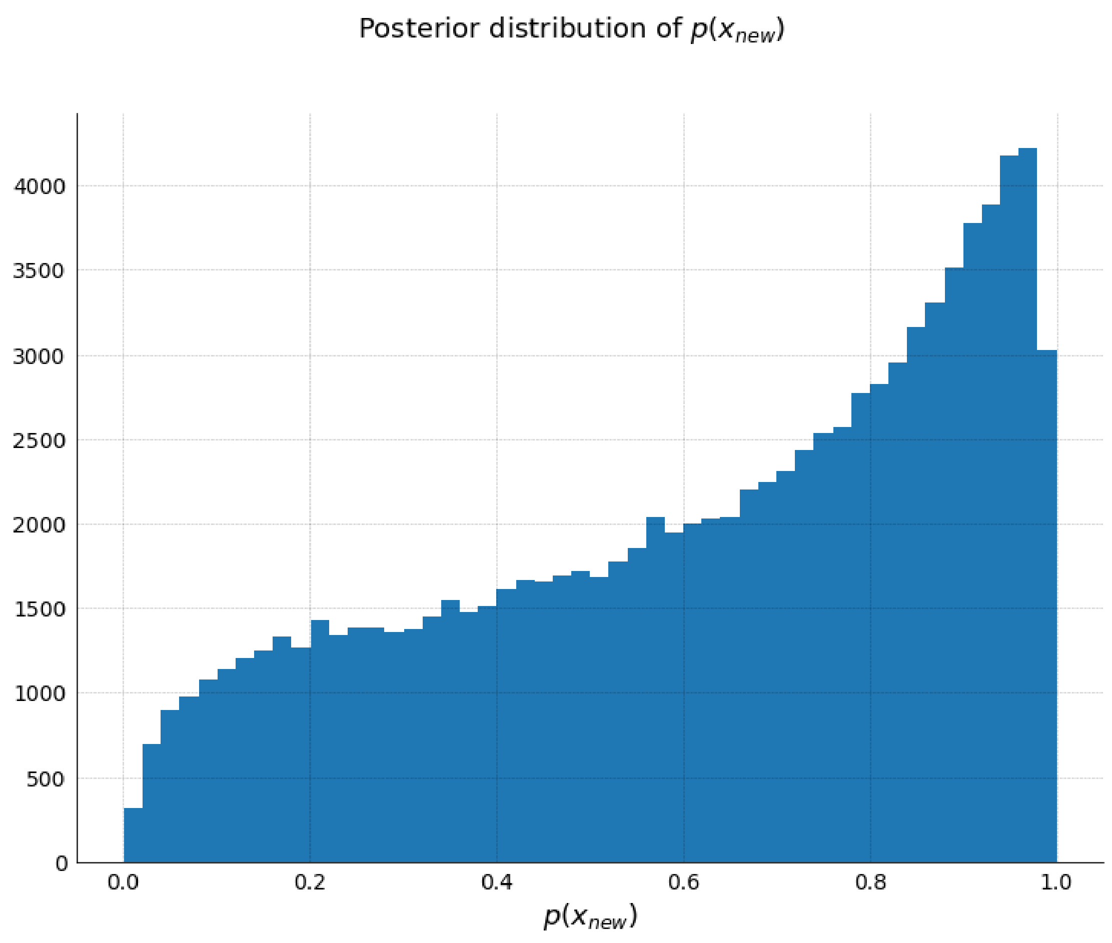

Figure 5.2: The posterior distribution of $p(x_{\text{new}})$ for the high dosage level $x_{\text{new}} = 2.50$.

to

```
if -8.0 <= a <= 8.0 and 0.0 <= b <= 8.0:
```

The result of MCMC with new interval for a and b is:

```
Out[8]:
----------------------
MCMC Mean:
-- a: -1.282
-- b: 0.834
----------------------
MCMC Std:
-- a: 0.575
-- b: 0.441
```

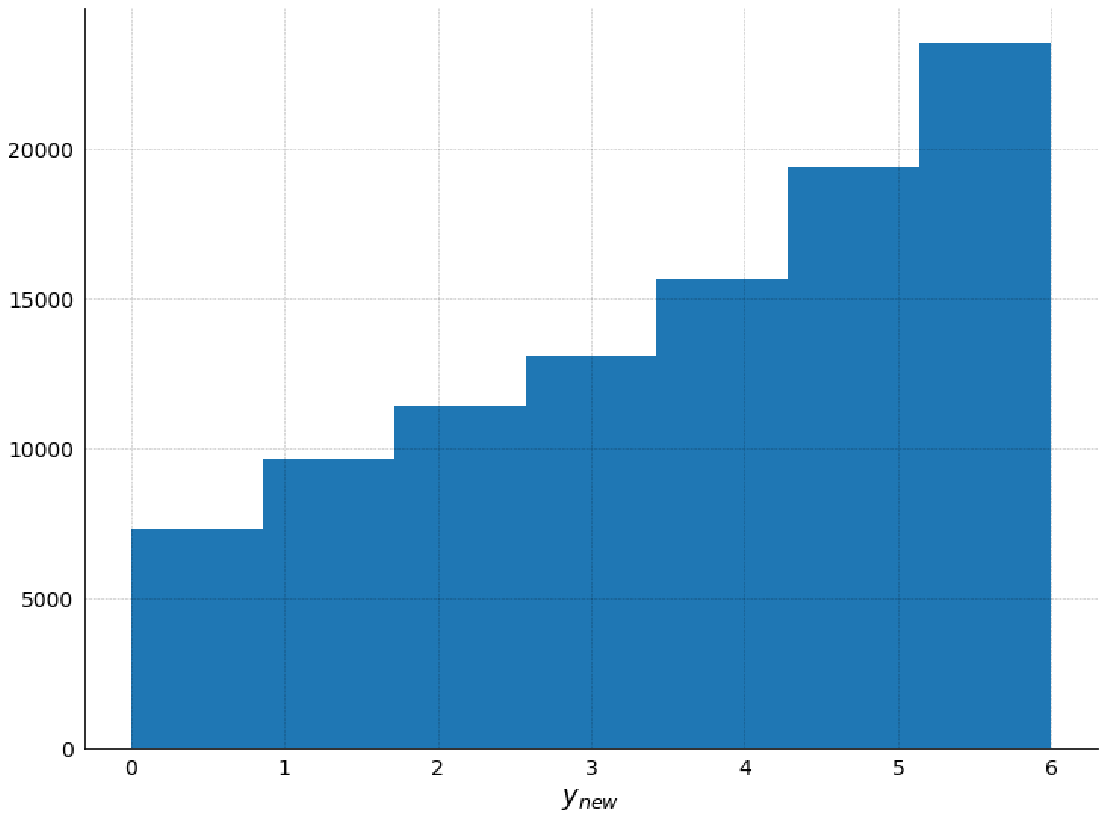

Figure 5.3: The predictive distribution for y_{new}, the number of $m = 6$ Ratti norvegici experiencing Event A when the dosage is $x_{\text{new}} = 2.50$.

The LD50 parameter is for $p = 50\%$, so I need to find the reverse of (5.2), i.e. x:

$$p = \frac{1}{1+e^{-z}} \quad \to \quad z = \ln\frac{p}{1-p}, \quad \text{where} \quad z = a + bx, \tag{5.8}$$

Therefore

$$x = \frac{\ln\frac{p}{1-p} - a}{b}, \tag{5.9}$$

Remembering $p = 1/2$, I get

$$x = -\frac{a}{b}, \tag{5.10}$$

This results in the following code for quantiles calculation of LD50 point:

```
samples_50 = []
for i in range(Nsamples):
    # Sampling new values for a and b to find LD50 point
    # We use Gaussian because number of samples is high
    a_50 = np.random.normal(a_mcmc_mean2, a_mcmc_std2)
    b_50 = np.random.normal(b_mcmc_mean2, b_mcmc_std2)
    # Cutting
    while b_50 < 0:
        b_50 = np.random.normal(b_mcmc_mean2, b_mcmc_std2)

    samples_50.append(-a_50/b_50)

print('LD50␣quantiles:')
print(f"--␣0.05:␣{np.quantile(samples_50,␣0.05):0.3f}")
print(f"--␣0.50:␣{np.quantile(samples_50,␣0.50):0.3f}")
print(f"--␣0.95:␣{np.quantile(samples_50,␣0.95):0.3f}")
```

Output is

```
Out[12]:
LD50 quantiles:
-- 0.05: 0.348
-- 0.50: 1.501
-- 0.95: 6.531
```