Bayesian Inference on Single-Parameter Models

Nikolas Tsigkas

April 2021

1 Prior to Posterior Update for Bernoulli Trials

The model for the trial data is that $Y_i|\theta \overset{iid}{\sim} Bern(\theta)$. If the chosen distribution for the prior is $\theta \sim Beta(\alpha_0, \beta_0)$, then the posterior distribution will be $\theta|y_1 ... y_n \sim Beta(\alpha_0 + s, \beta_0 + f)$ where s and f denote the number of successes and failures (ones and zeros) in our sample data. This is because the Beta distribution is a conjugate prior to the Bernoulli distribution.

With $\alpha_0 = \beta_0 = 3$, $s = 8$ and $n = 24$, i.e. $f = 16$, the analytical mean and standard deviation of the posterior distribution is $\mu = 0.3666...$ and $\sigma = 0.0865....$ The mean and standard deviation can also be evaluated by drawing random samples from the posterior distribution, given that the sample size is large enough. This is illustrated in figure 1, where one can see that the two values converge towards their analytically derived values.

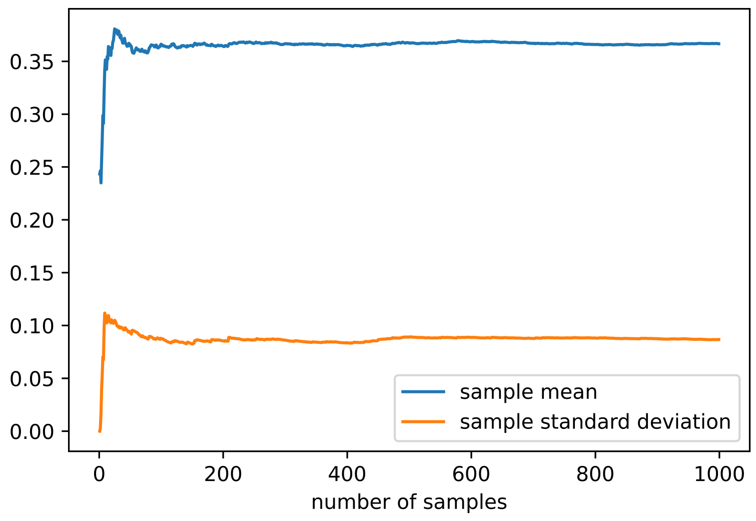

Figure 1: Convergence of the sample mean and standard deviation

From the samples, one can also evaluate probabilities for the parameter θ. For instance $P[\theta > 0.4|y_1..y_n] \approx \frac{n_{0.4}}{n} = 0.3445$ when $n_{0.4}$ denotes the number of samples greater than or equal to 0.4 for $n = 10000$ samples. This can be compared to the true probability, given the posterior distribution $1 - F(0.4) = 0.3426$ where F denotes the cdf of the posterior distribution.

Another way to illustrate the distribution of the posterior is to transform it to the log-odds: $\phi = log(\frac{\theta}{1-\theta})$. The histogram for ϕ is illustrated in figure 2, reusing the simulated draws from before.

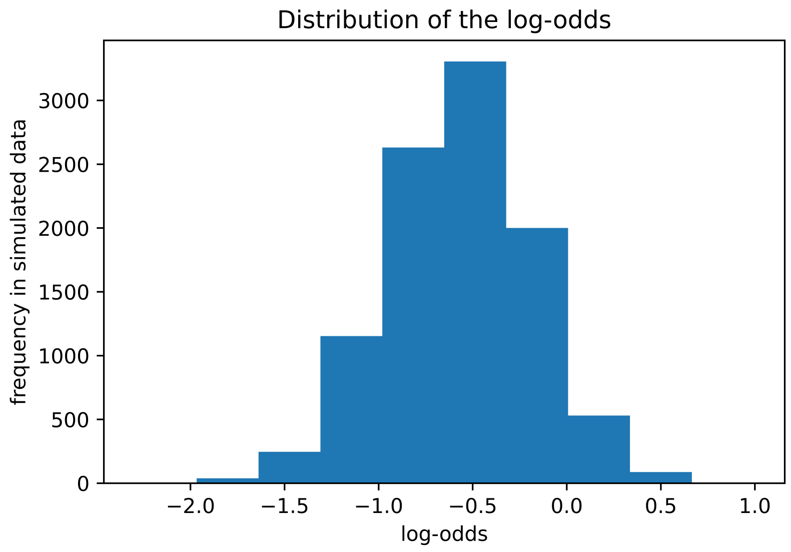

Figure 2: Histogram of the log-odds

```python
from scipy import stats as st
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(123) # set seed for reproducability

# Parameters for beta prior distribution
alpha0 = 3
beta0  = 3

n = 24 # number of samples from Bernoulli trial data
s = 8 # number of sucesses in sample
f = n - s # number of failiures

# Analytical mean and standard deviation for posterior
mean, var = st.beta.stats(alpha0+s, beta0+f, moments="mv")
std = np.sqrt(var)

# Draw 10 000 random samples from posterior
post_sim = st.beta.rvs(alpha0+s, beta0+f, size=10000)

# Check convergence of mean, standard deviation
mean_sim = np.zeros(1000)
std_sim  = np.zeros(1000)

for i in range(1000):
    mean_sim[i] = np.mean(post_sim[0:i])
    std_sim[i]  = np.std(post_sim[0:i])

plt.plot(mean_sim)
plt.plot(std_sim)
plt.legend(["sample mean","sample standard deviation"])
plt.xlabel("number of samples")
plt.savefig("mean_sd_convergence.png", dpi=1500)
plt.show

#Compare simulated probability to exact posterior
geq04 = post_sim >= 0.4
# ratio of number of samples >= 0.4 and total samples
p_sim = float(sum(geq04))/float(len(post_sim))
p_true = 1 - st.beta.cdf(0.4, alpha0 + s, beta0 + f)

# Explore the distribution of the log-odds
logodds_sim = np.log(post_sim/(1-post_sim))
plt.hist(logodds_sim)
plt.title("Distribution of the log-odds")
plt.xlabel("log-odds")
plt.ylabel("frequency in simulated data")
plt.savefig("logodds_hist", dpi = 1500)
plt.show
```

2 The Gini Coefficient and the log-normal distribution

The model used for the data of income levels is $Y_i|\mu,\sigma \overset{iid}{\sim} \log N(\mu,\sigma^2)$, or equivalently $log(Y)|\mu,\sigma \overset{iid}{\sim} N(\mu,\sigma^2)$. Assuming that $\mu = 3.8$ is known, and using a non informative prior for σ, i.e. $p(\sigma^2) \propto 1/\sigma^2$, the posterior distribution will be $\sigma^2|y_1,..,y_n,\mu \sim$ Scale-Inv-$\chi^2(n,\tau^2)$, where τ^2 is defined as:

$$\tau^2 = \frac{\sum_1^n (ln(y_i) - \mu)^2}{n}$$

To simulate from this distribution, the algorithm is the following:

1. Draw n samples from $X \sim \chi^2(n-1)$
2. Calculate τ^2 using the data and the formula above
3. Calculate $\sigma^2 = \frac{(n-1)\tau^2}{X}$
4. The result is $\sigma^2 \sim$ Scale-Inv-$\chi^2(n,\tau^2)$

If the population income follows a lognormal distribution, then the Gini coefficient takes on the value $G = 2\Phi(\sigma/\sqrt{2})$ where Φ is the cdf of the standard normal. By drawing samples of σ following the algorithm above, we can explore the distribution of the Gini coefficient. Figure 3 illustrates a histogram of the simulated Gini coefficients.

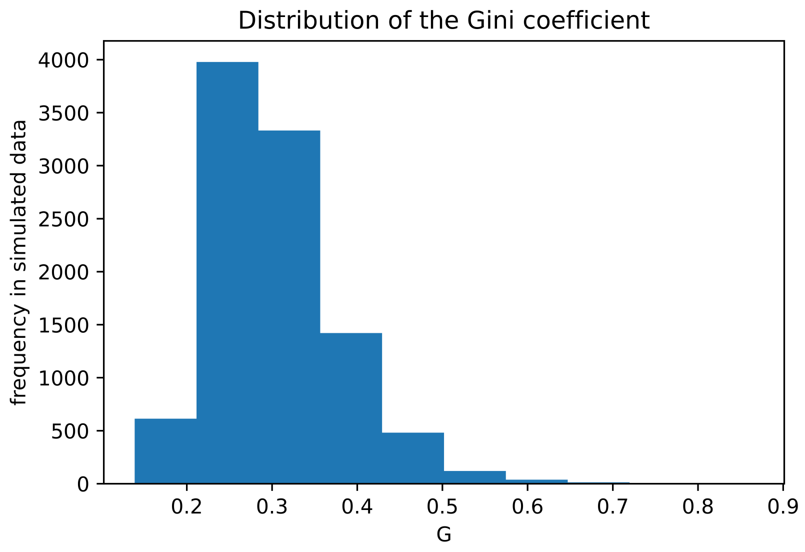

Figure 3: Histogram of the Gini coefficient

From the simulated data, one can also fit a density kernel to explore further properties of the posterior distribution, such as the equal tail credible interval and the highest posterior density interval (HPDI). The resulting intervals, on a 90% level, were:

$$\begin{aligned} HPDI_{0.9} &= [0.1892; 0.4184] \\ CI_{0.9} &= [0.2052; 0.4464] \end{aligned}$$

One can see that allthough both intervals contain 90% of the probability mass, they are not equal, which follows from the fact that the distribution is skewed. The fitted density, together with the corresponding intervals, is shown in figure 4.

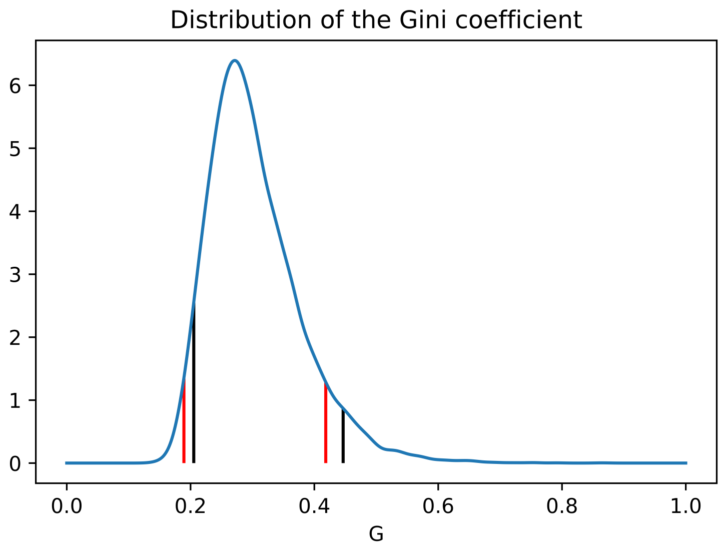

Figure 4: Density, equal tail credible interval (black) and HPDI (red) for the Gini distribution

```python
from scipy import stats as st
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import quad

np.random.seed(123) # set seed for reproducability

# Dataset of monthly income in KSEK/mo.
y = np.array([38, 20, 49, 58, 31, 70, 18, 56, 25, 78])
n = len(y)

# Known parameter of log-normal distribution
mean = 3.8

# Draw 10 000 random samples from the scaled inverse chi2
# by first drawing from a chi2 distribution with n-1 df's
tau2 = np.sum((np.log(y)-mean)**2)/n
X = st.chi2.rvs(n-1, size=10000)
var_sim = (n-1)*tau2/X

# Explore the distribution of the Gini-coefficient
# By fitting a gaussian density kernel to the data
G = 2*st.norm.cdf(np.sqrt(var_sim/2))-1
G_pdf = st.gaussian_kde(G)

def G_cdf(x):
    return quad(G_pdf, 0, x)[0]

# Plot histogram from the simulated variances
plt.hist(G)
plt.title("Distribution of the Gini coefficient")
plt.xlabel("G")
plt.ylabel("frequency in simulated data")
plt.savefig("gini_hist.png", dpi=1500)
plt.show

# Evaluate the pdf and cdf for a dense grid of values.
x = np.linspace(0,1,1000)
G_pdf_vals = G_pdf(x)
G_cdf_vals = np.zeros_like(x)

for i, vals in enumerate(x):
    G_cdf_vals[i] = G_cdf(vals)

# Generate 90% highest posterior density interval
sorted_index = np.argsort(G_pdf_vals)[::-1]
HPDI = [x[sorted_index[1]]]

# By adding points until they integrate to 0.9
for i in range(1,len(sorted_index)):
    if quad(G_pdf,min(HPDI), max(HPDI))[0] < 0.9:
        HPDI.append(x[sorted_index[i]])

HPDI = [min(HPDI), max(HPDI)]

# Generate 90% equal tail credible interval from the cdf values
# By having 5% of the mass to the left and to the right
lb_index = np.argmin(np.abs(G_cdf_vals-0.05))
ub_index = np.argmin(np.abs(G_cdf_vals-0.95))

eq_tail_CI = [x[lb_index], x[ub_index]]

# Plot density
# With the equal tail CI and HPDI
plt.plot(x,G_pdf_vals)
plt.title("Distribution of the Gini coefficient")
plt.xlabel("G")
plt.vlines(eq_tail_CI[0], 0, G_pdf(eq_tail_CI[0]))
plt.vlines(eq_tail_CI[1], 0, G_pdf(eq_tail_CI[1]))
plt.vlines(HPDI[0], 0, G_pdf(HPDI[0]), color = "red")
plt.vlines(HPDI[1], 0, G_pdf(HPDI[1]), color = "red")
plt.savefig("gini_density.png", dpi = 1500)
plt.show
```

3 Bayesian Inference on the von-Mises distribution

The model used for the data of wind directions (in radians) is $Y_i|\mu, \kappa \overset{iid}{\sim}$ von-Mises(μ, κ). The pdf of this distribution is:

$$\frac{e^{(\kappa \cos(y-\mu))}}{2\pi I_0(\kappa)} \quad \forall - \pi \leq y \leq \pi$$

Where I_ν is the modified Bessel function of the first kind of order ν. By assuming $\mu = 2.39$ to be known, and using a prior distribution for $\kappa \sim Exp(\lambda)$, we can derive the posterior through:

$$p(\kappa|y_1, .., y_n, \mu) \propto \lambda e^{-\lambda\kappa} \prod_1^n \frac{e^{\kappa \cos(y-\mu)}}{2\pi I_0(\kappa)} \propto (\frac{\lambda}{I_0(\kappa)})^n \exp[\kappa(\sum_1^n \cos(y_i - \mu) - \lambda)]$$

To make this a propper pdf (i.e. that it integrates to one), it is normalized by dividing the expression above with its integral. The resulting distribution when $\lambda = 1$ is plotted in figure 5.

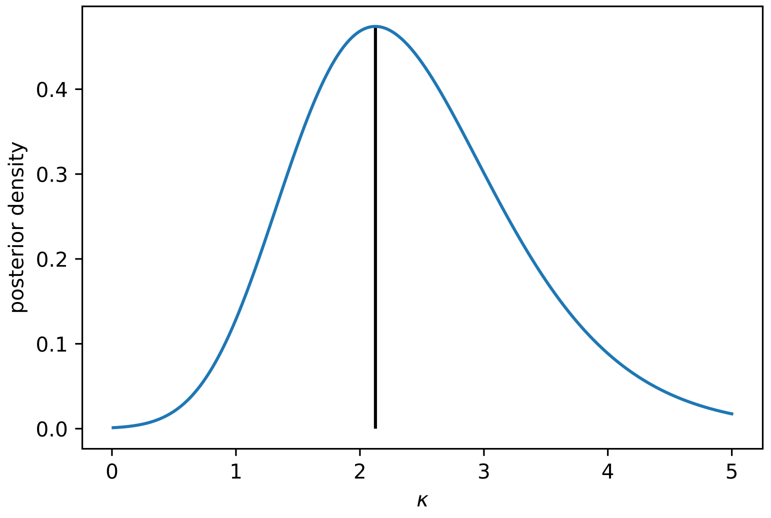

Figure 5: Posterior distribution of κ

Finally, the approximate posterior mode is found, as the value of κ that yields the highest pdf-value, as $\kappa = 2.1249$, which is also highlighted by the vertical line in the plot above.

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import iv as bessel
from scipy.integrate import quad

np.random.seed(123) # set seed for reproducability

# Dataset of wind directions in radians
y = np.array([-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02])

mu    = 2.39
lamda = 1

# Function proportional to the posterior distribution
# Given von-Mises(kappa,mu) likelihood and Exp(lamda) prior
def posterior(kappa, y, lamda, mu):
    n = len(y)
    return (lamda/bessel(0,kappa))**n*np.exp(kappa*(np.sum(np.cos(y-mu))-lamda))

# Find posterior for different values of kappa
kappa = np.linspace(0.01,5,10000)
posterior_kappa = np.zeros_like(kappa)

posterior_kappa = posterior(kappa, y, lamda, mu)

# Normalize the posterior so that it integrates to 1
posterior_kappa = posterior_kappa/(quad(posterior, 0.01, 5, args = (y, lamda, mu))[0])

# Find the posterior mode from the density
mode = kappa[np.argmax(posterior_kappa)]

# Plot posterior
plt.plot(kappa, posterior_kappa)
plt.xlabel(r"$\kappa$")
plt.ylabel("posterior density")
plt.vlines(mode,0,np.max(posterior_kappa))
plt.savefig("kappa_posterior_pdf.png", dpi=1500)
plt.show
```