UNIVERSITY OF
CAMBRIDGE

MPhil in Data Intensive Science

S2 - Statistics for Data Science

The Lighthouse Problem

CRSid: yz870

Date: 28th March 2024

word count ≈ 2600

Table of Contents

List of Figures

List of Tables

Lighthouse Problem with Flash Locations

A lighthouse stands at position α along a linear coastline and extends a distance β out into the sea. It rotates and emits flashes at angles θ, uniformly distributed. The light beams are narrow and intersect the coastline at a single point when the angle falls within the range of $-\frac{\pi}{2}$ to $\frac{\pi}{2}$. Detectors arranged along the coastline register the locations x_k (where $k = 1, 2, ..., N$) where N flashes are detected. These detectors only indicate the occurrence of a flash but not the intensities of the light. This section aims to construct a Bayesian model with given information to find the location of the Lighthouse.

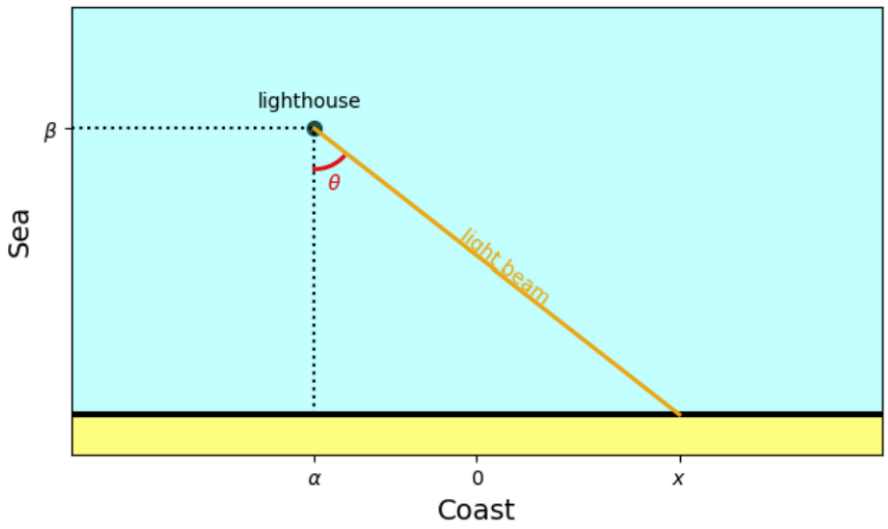

Figure 1: Setup of the lighthouse problem

i) & ii) Define likelihood function

From Figure 1, we can see a right-angled triangle with β ($\beta > 0$ by definition) and $|x - \alpha|$ as its legs. If we define $0 < \theta < \frac{\pi}{2}$ when x is on the right side of α and $-\frac{\pi}{2} < \theta < 0$ when x is on the left side of α, then the trigonometric relationship between α, β, θ, and x is:

$$tan(\theta) = \frac{x - \alpha}{\beta} \tag{1}$$

equivalently,

$$\theta = arctan(\frac{x - \alpha}{\beta}) \tag{2}$$

From the problem description, we know θ follows $Uniform(-\frac{\pi}{2}, \frac{\pi}{2})$ distribution, i.e.

$$f(\theta) = \begin{cases} \frac{1}{\pi} & \text{for } \theta \in (-\frac{\pi}{2}, \frac{\pi}{2}) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

From equation (1), if we define $h(\theta) = \beta \times tan(\theta) + \alpha$, then $x = h(\theta)$. Since h is a monotonically increasing and invertible function of θ, we can define $\theta = h^{-1}(x)$ as shown

in equation (2), and find the PDF of x as follows:

$$\begin{aligned} L_x(x|\alpha,\beta) &= f(h^{-1}(x))\left|\frac{d\theta}{dx}\right| \\ &= \frac{1}{\pi}\times\frac{1}{1+(\frac{x-\alpha}{\beta})^2}\times\frac{1}{\beta} \quad \text{since } \frac{dtan^{-1}(u)}{du}=\frac{1}{1+u^2} \\ &= \frac{\beta}{\pi(\beta^2+(x-\alpha)^2)} \quad \forall x\in\mathbb{R} \end{aligned} \tag{4}$$

This is, by definition, a *Cauchy*, or *Lorentzian* distribution. Since all the flashes are emitted at independent angles, detected locations x_ks are independent and thus the likelihood $L_x(\{x_k\}|\alpha,\beta)=\prod_{k=1}^{N}L_x(x_k|\alpha,\beta)$

iii) Find a "good" estimator of α

In the PDF of x we deduced above (Eqn.4), we can notice that the denominator $(\beta^2+(x-\alpha)^2\geq\beta^2$ and thus $L_x(x_k|\alpha,\beta)\leq\frac{\beta}{\pi\beta^2}=\frac{1}{\pi\beta}$. The equality holds true at $x=\alpha$, hence the most likely location, where L is maximised, for a single flash to be received is $x=\alpha$.

However, this doesn't necessarily indicate that the sample mean $\overline{x}=\frac{1}{N}\sum_{k=1}^{N}x_k$ is the best estimator of α. In fact, since the *Cauchy* distribution has no well-defined moments, the parameters α and β do not correspond to any moments, so attempting to estimate them from sample moments will very possibly not succeed (Lohninger 2017).

Computation for the maximum likelihood estimate of α and β starts with computing the log-likelihood $l_x(x_k|\alpha,\beta)$ and setting its derivatives with respect to α and β to be 0:

$$\frac{\partial l}{\partial\beta}=\frac{N}{\beta}-\sum_{k=1}^{N}\frac{2\beta}{\beta^2+(x_i-\alpha)^2}=0 \;\&\; \frac{\partial l}{\partial\alpha}=\sum_{k=1}^{N}\frac{2(x_i-\alpha)}{\beta^2+(x_i-\alpha)^2}=0 \tag{5}$$

The solution of these equations requires finding the roots of high-degree polynomials, which is nearly impossible to do analytically. Using the recorded 20 locations in the given dataset *lighthouse flash data.txt*, we numerically compute the MLE by minimizing negative log-likelihood, and then compare with sample mean and medium.

Table 1: Comparisons between estimates of α

maximum likelihood estimate	**sample mean**	**sample median**
-0.52219	3.10831	-0.42156

From the above table, the sample mean deviates from the MLE, a good estimator, much more than the sample medium, and thus is NOT a good estimator for α.

iv) Choose priors for α and β

We assume α and β are independent, and since β is always positive and measures the distance, we want it to be invariant under change of the units. We thus choose a log-uniform prior with support $(0.01, 5)$ on β. For α, its maximum likelihood estimate (MLE) is approximately zero. Therefore, we incorporate a normal prior with a mean of 0 and a variance of 4. This choice reflects the high probability of α being close to zero while allowing for some uncertainty in its precise value.

v) Draw stochastic samples from 2-d posterior distribution

Using recorder lighthouse flash locations x_k $k = 1, 2, 3, ..., 20$ and the priors we assumed, we then can run Markov Chain Monte Carlo (MCMC) simulations to draw samples from the posterior distribution of α and β $P(\alpha, \beta|\{x_k\})$. In Bayesian statistics, what we are interested in and what most MCMC algorithms and software packages require is the unnormalized posterior $P^*(\alpha, \beta|\{x_k\})$ (equation 6). We also draw contours for P^* to get some hint of the appropriate MCMC method that can be adopted here.

$$\begin{aligned} P^*(\alpha, \beta|\{x_k\}) &\propto \pi(\alpha, \beta) L_x(\{x_k\}|\alpha, \beta) \\ &\propto 1_{(0.01,5)}(\beta) e^{\frac{\alpha^2}{-2\times 4^2}} \frac{\beta^{20}}{\prod_{k=1}^{20}(\beta^2 + (x_k - \alpha)^2)} \end{aligned} \tag{6}$$

where $1(x)$ is the indicator function.

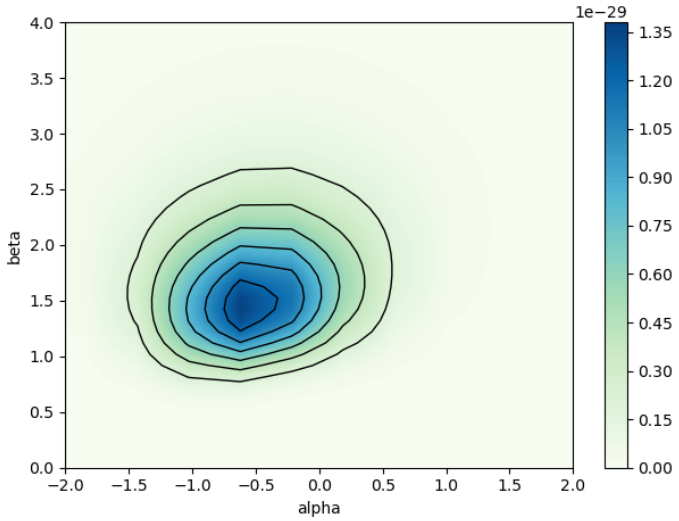

Figure 2: Unnormalized posterior distribution of α and β

From the plot above, it's evident that the posterior distribution closely resembles a 2D Gaussian distribution. This characteristic makes it relatively straightforward to sample from and poses minimal challenges for the sampling algorithm. Consequently, we initiate our exploration with the simplest MCMC method: Metropolis-Hastings. The results obtained from this approach appear satisfactory (see Diagnostics section).

Method: Metropolis-Hasting

Metropolis-Hasting (MH) algorithm can iteratively generate a sequence of samples x_i from the target distribution P^* by proposing new states y from the proposal distribution $Q(y|x)$ and deciding whether or not to accept the new state based on acceptance ratio $\frac{P^*(y)Q(x_i|y)}{P^*(x_i)Q(y|x_i)}$. The most basic version of MH proceeds as follows,

Algorithm 1 Metropolis-Hastings Algorithm

Initialize x_0
for $i = 0, 1, 2, \ldots$ **do**
  Generate a proposal y from the proposal distribution $Q(y|x_i)$
  Compute the acceptance ratio:
    $r = \frac{P^*(y)}{P^*(x_i)} \cdot \frac{Q(x_i|y)}{Q(y|x_i)}$
  Generate a uniform random number $u \sim U(0, 1)$
  if $u < r$ **then**
    Accept the proposal: $x_{i+1} = y$
  else
    Reject the proposal: $x_{i+1} = x_i$
  end if
end for

While the MH algorithm is capable of eventually converging to the target distribution P with any choice of a proposal distribution Q, it often exhibits undesirable random walk behavior. This can result in slow diffusion around the sample space, requiring a large number of iterations to explore effectively. To improve the performance of MH sampling, it is essential to select a proposal distribution that closely matches the target distribution and tune the acceptance ratio to around 30%. In our case, we opt for a 2D Gaussian proposal distribution Q with a covariance matrix being the identity matrix.

In addition, we opt to use the logarithm of the posterior rather than the posterior itself as the target distribution to simplify computations and improve numerical stability. It takes 11.16 seconds to run 10000 steps and the overall acceptance ratio is 32.4%.

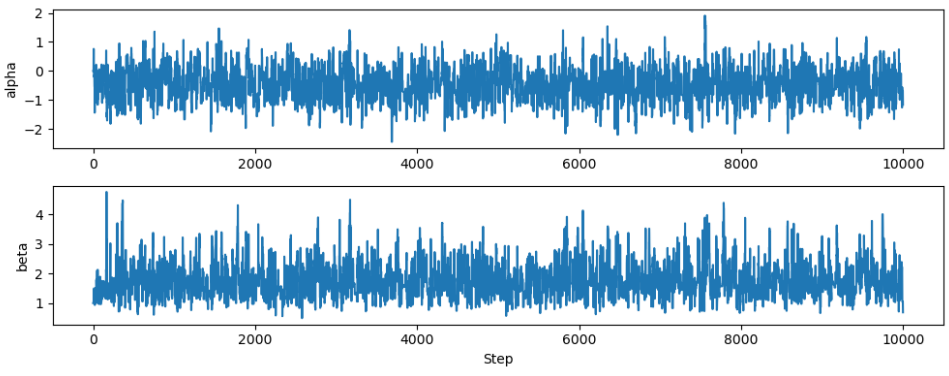

Figure 3: Generated α (above) and β (below) chains from MH

Diagonistics

We systematically evaluate the quality and reliability of samples generated from MH algorithm following the diagnostics steps listed below.

1. **Visual Inspection** In Figure 3 above, it's not immediately apparent whether there's a presence of burn-in, an early proportion of the chain that may have not converged to target distribution yet, though periodical peaks in the chain may suggest there's autocorrelation between different sections of the chains. To further investigate, we plot the first 150 steps of the chain in 2D parameter space, as shown in Figure 4. From this visualization, we observe that while the chain exhibits some "random walk" properties, it in general effectively explores the parameter space and doesn't appear to be trapped in local modes.

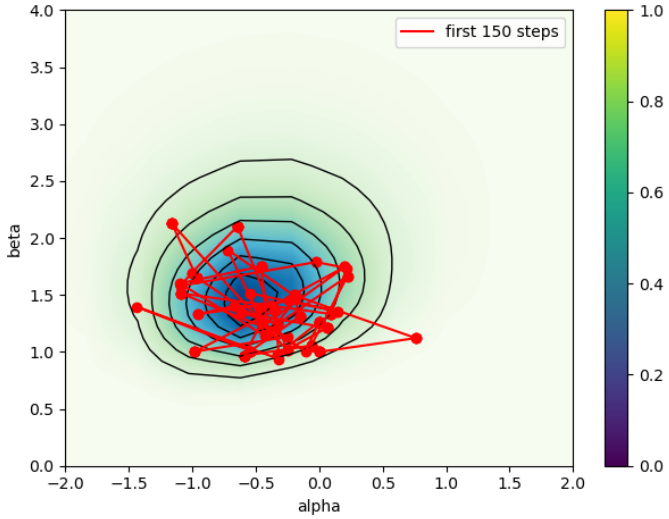

Figure 4: First 150 steps in the 2d parameter space

2. **Geweke Diagnostic** To rigorously identify any potential burn-in period requiring removal, we apply Geweke tests (Geweke 1992) to compare different initial proportions with the last 50% of the chain which is convergent to the target distribution by assumption.

   According to Geweke 1992, Geweke test mimics a basic 2-sample test of means but utilizes spectral densities to estimate sample variances. Using a significance level of 5%, we regard the earliest proportion of the samples exhibiting a z-score with an absolute value exceeding 1.96 as the burn-in period. Since the Geweke test was originally designed for the univariate case, we take the maximum of burn-in detected from α and β components, which is 789, and discard.

3. **Gelman-Rubin statistic** We delve deeper into assessing the convergence of the chain using the Gelman-Rubin Test (Gelman and Rubin 1992), a method that evaluates MCMC convergence by comparing multiple Markov chains. The convergence

is evaluated using the potential scale reduction factor (PSRF), which estimates the potential reduction in between-chains variability relative to within-chain variability, with a favorable value approaching 1 (Gelman and Rubin 1992).

To perform the test, we rerun an additional 5 chains and compute the PSRF for α and β after discarding the burn-in period. These results yield PSRF values of 1.0006 and 1.0004, respectively. This close proximity to 1 indicates consistent sampling results produced by the Metropolis-Hastings algorithm, suggesting a high likelihood of convergence.

4. **Autocorrelation Analysis** To ensure the reliability of estimates and their uncertainties from MCMC simulations, it's crucial to find independent samples from the original chain. The integrated autocorrelation time (IAT) is a key metric used to assess the level of autocorrelation within the chain. It is computed by summing the autocorrelation function over all lags and estimates the total number of steps required for the chain to produce one effectively independent sample. In our analysis, we set the integrated autocorrelation time of the chain as the maximum of the IAT values for its components, approximately equal to 8.6.

5. **Effective Sample Size** After discarding the burn-in period, we conservatively extracted independent samples from the chain every 2 times the IAT, resulting in a total of 576 samples. Analysis revealed that 5.76% of the generated samples were deemed effective. Notably, each effective sample was generated in approximately 0.019 seconds, which demonstrates good performance for the MH algorithm.

Results

After thinning the Markov chain to 576 effective samples, we generate plots of histograms of the posterior marginal and joint distributions of α and β, as shown in Figure 5. The mean and standard deviation of these parameters are presented in Table 2.

Table 2: Mean and standard deviation of posterior α and β

	mean	variance
α	-0.45487	0.56347
β	1.75326	0.56873

The posterior means of α and β are close to their MLEs which are approximately -0.5 and 1.6 respectively. This result is consistent with the ignorant nature of the priors we choose.

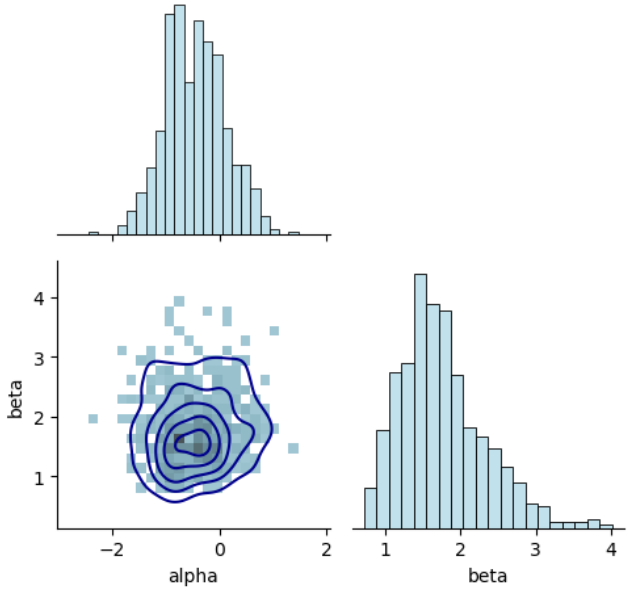

Figure 5: Marginal and joint posterior distributions of α and β

Back to the real-world description of the problem, (α, β) is the coordinate of the lighthouse, hence we found the possible region of its location as shown in the bottom left corner in Figure 5 with the deepness of colour representing the probability of exactly located at that point.

Lighthouse Problem with Flash locations & Intensities

In the following analysis, not only the locations $\{x_k\}$ $k = 1, 2, 3, ...20$ are known, but the flash intensities $\{I_k\}$ $k = 1, 2, 3, ...20$ at these locations are also recorded. These intensity measurements are independent and follow a log-normal distribution with PDF:

$$L_I(logI|\alpha, \beta, I_0) = \frac{exp(\frac{-(logI-\mu)^2}{2\sigma^2})}{\sqrt{2\pi\sigma^2}} \tag{7}$$

where uncertainty $\sigma = 1$ and expectation $\mu = log(\frac{I_0}{d^2})$, in which $d^2 = \beta^2 + (X - \alpha)^2$ is the squared distance between the detector and lighthouse and I_0 is the absolute intensity of the lighthouse. It is assumed that the recorded xs and Is are independent thus the model likelihood becomes:

$$\begin{aligned} L_{x,I}(\{x_k\}, \{logI_k\}|\alpha, \beta, I_0) &= L_x(\{x_k\}|\alpha, \beta)L_I(\{logI_k\}|\alpha, \beta, I_0) \\ &= \prod_{k=1}^{N} L_x(x_k|\alpha, \beta)L_I(logI_k|\alpha, \beta, I_0) \end{aligned} \tag{8}$$

vi) choose priors for I_0

Before incorporating new intensity data into our Bayesian model, we must specify the prior distribution for the new parameter I_0. The following 3 aspects of I_0 are considered:

- I_0 represents the intensity of light at the lighthouse and must be greater than any observed I_k, so the domain of I_0 is constrained to always be within $(max(I_k), \inf)$.
- The expectation of I_k, denoted as μ_k, equals $log(I_0) - log(d^2)$, hence we anticipate that $log(I_0)$ follows a distribution similar to the expectation of a log-normally distributed variable.
- I_0 is more likely to fall within a range slightly larger than $max(I_k)$ rather than being significantly further away.

Based on the above characteristics of I_0, we assume $I_0 - max(I_k)$ follows a log-normal distribution with mean $\mu_0 = 2$ and $\sigma_0 = 1$. Figure 6 shows the PDF of the prior distribution of I_0, where the red dashed line is the maximum intensity observed.

Finally, we assume I_0, α, and β are mutually independent, and thus the joint prior is the product of three individual priors and the unnormalized posterior becomes:

$$\begin{aligned} P^*(I_0, \alpha, \beta|\{x_k\}, \{I_k\}) &\propto \pi(I_0, \alpha, \beta)L_{x,I}(\{x_k\}, \{logI_k\}|\alpha, \beta, I_0) \\ &\propto 1_{(0.01,5)}(\beta)exp(\frac{\alpha^2}{-2 \times 4^2} - \frac{(logI_0 - 2)^2}{2}) \\ &\quad \times \frac{\beta^{20}}{\prod_{k=1}^{20}(\beta^2 + (x_k - \alpha)^2)} exp(\frac{-\sum_{k=1}^{20}(logI_k - \mu_k)^2}{2}) \end{aligned} \tag{9}$$

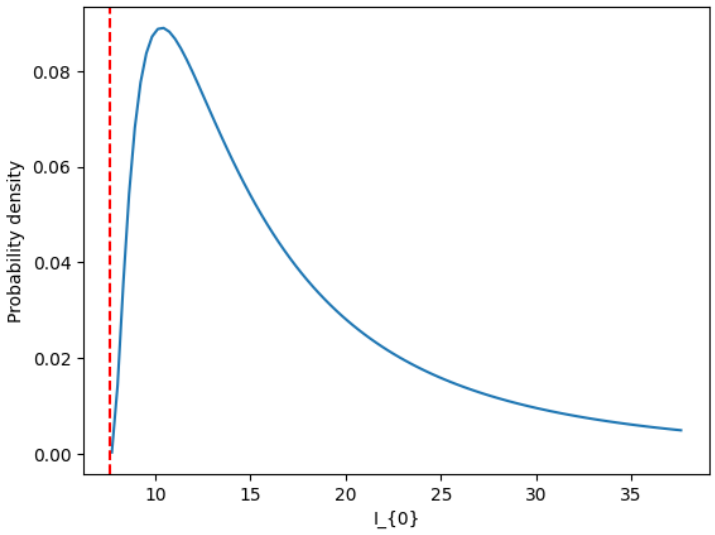

Figure 6: PDF of prior distribution of I_0

vii) Draw stochastic samples from 3-d posterior distribution

We proceed by repeating part **v)** to draw stochastic samples from $P^*(I_0, \alpha, \beta|\{x_k\}, \{I_k\})$. However, the right-skewness inherent in the prior distribution of I_0 complicates the joint prior, rendering it less Gaussian-like. Consequently, selecting an appropriate proposal distribution in Metropolis-Hastings (MH) sampling becomes challenging. To circumvent this issue, we opt for emcee, a Python implementation of Affine Invariant MCMC Ensemble sampler (Foreman-Mackey et al. 2013), due to its fast sampling of skewed distributions.

Package: emcee

An affine transformation of X takes the form $Y = AX + b$, where A is invertible, is possible to convert a challenging sampling problem on X into an easy one on a well-scaled Y. Accoding to Goodman and Weare 2010, an affine invariant sampler treats these difficult and easy sampling densities as equally challenging, making it particularly useful for scenarios where traditional sampling methods encounter difficulties.

In this context, emcee implements an ensemble of affine invariant samplers, referred to as "walkers" (Foreman-Mackey et al. 2013). These walkers collectively explore the parameter space, with the target probability density being independent and drawn from the specified density. This approach allows for efficient exploration of complex parameter spaces by adapting to their local geometries, thereby improving sampling efficiency and convergence properties. We use 10 independent walkers starting from random (I_0, α, β) values, each running 10000 steps to draw samples from P^*.It takes about 73 seconds and the acceptance fraction is around 63%.

Diagonistics & Results

We followed the same steps as we have done for the diagnostics of 2d samples in part **v)**.

1. **Visual Inspection** Below are plots of 10 chains, each comprising 3 components, generated from the emcee. It appears that the Markov chains rapidly move away from their initial positions, indicating a potential convergence to the target distribution. However, there are noticeable autocorrelations at some lag values within the chains.

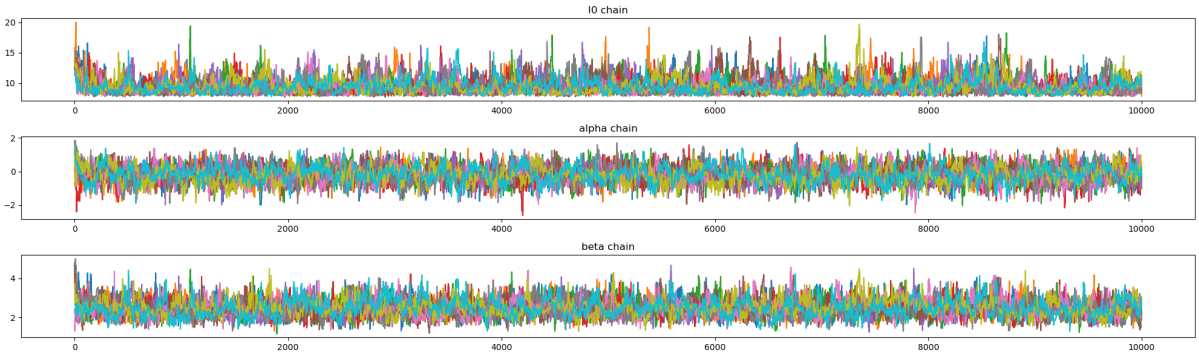

Figure 7: 10 independent chains for I_0 (top), α (middle), and β (bottom)

2. **Geweke Diagnostic** We perform Geweke tests for each univariate chain in the ensemble, and it turns out the suggested burn-in period lengths are all similar, we thus regard their average as final burn-in (307) to be discarded.

3. **Gelman-Rubin statistic** Gelman-Rubin tests computed for I_0, α, and β from an ensemble of 10 chains are 1.00571004, 1.0075218, and 1.00488354 respectively, suggesting good consistency between different walkers.

4. **Autocorrelation Analysis** The maximum of the estimates of the autocorrelation time for each parameter is about 52.93.

5. **Effective Sample Size** After removing the first 307 samples from all chains, we extracted 1860 effective samples from the chains at intervals of twice the Integrated Autocorrelation Time (IAT). This process yields a total of 18.6% effective samples from the generated dataset and it take 0.039 seconds to obtain each effective sample.

After thinning the chains into 1860 independent samples, we generate plots of histograms of the posterior marginal and joint distributions of I_0, α, and β as shown in Figure 8. The mean and standard deviation of these parameters are presented in Table 3.

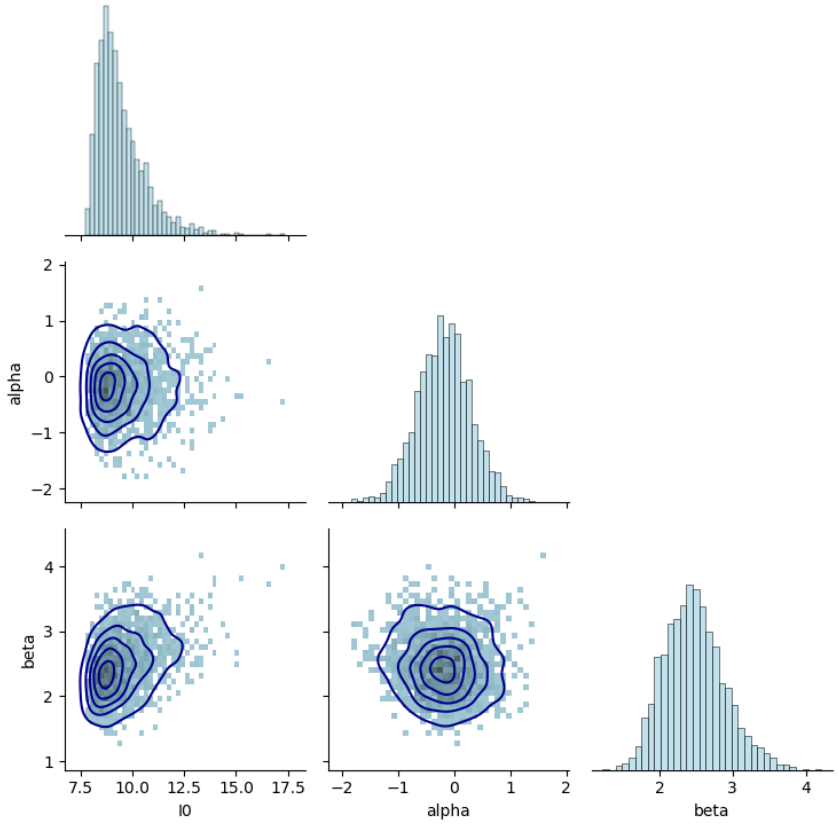

Figure 8: Marginal and joint posterior distributions of I_0, α and β

Table 3: Mean and standard deviation of posterior I_0, α and β

	mean	variance
I_0	9.44321	1.18505
α	-0.18387	0.48833
β	2.47164	0.41741

Comparing these results with the ones in Table 2, we found both the variance of α and β are reduced. This aligns with the intuitive understanding that increased data availability leads to more precise or accurate parameter estimates. Moreover, the mean of α is shifted towards 0, while the mean of β exhibits an opposite trend.

Furthermore, the posterior distribution of I_0 bears a striking resemblance to the prior distribution, as illustrated in Figure 6. This consistency suggests that our selection of prior aligns well with the observed patterns in the data.

References

Foreman-Mackey, Daniel et al. (2013). 'emcee: the MCMC hammer'. In: *Publications of the Astronomical Society of the Pacific* 125.925, p. 306.

Gelman, Andrew and Donald B Rubin (1992). 'Inference from iterative simulation using multiple sequences'. In: *Statistical science* 7.4, pp. 457–472.

Geweke, John (1992). 'Evaluating the accuracy of sampling-based approaches to the calculations of posterior moments'. In: *Bayesian statistics* 4, pp. 641–649.

Goodman, JONATHAN and JONATHAN Weare (2010). 'Communications in Applied Mathematics and Computational Science'. In: *Math. Sci. Publ.*

Lohninger, H. (Mar. 2017). URL: http://www.statistics4u.info/fundstat_eng/ee_distri_cauchy.html.

Appendix

A README

This project contains the codes used for solving the lighthouse problem described in lighthouse.pdf and reproducing all the results and figures shown in the report.

Installation

To install this project, follow these steps:

1. Clone from the Gitlab Repository

```
$ git clone https://gitlab.developers.cam.ac.uk/phy/data-intensive-science-mphil/S2_Assessment/yz870.git
```

2. Create and activate new conda environment

```
$ cd yz870
$ conda env create -f environment.yml
$ conda activate cw_s2
```

Usage

There 2 two main scripts, called *main_part1.py* and *main_part2.py*, containing solutions to questions i to v and solutions to questions vi to viii respectively. To run them on the command line, you need to specify the number of steps for MCMC simulations to run after –nsteps, e.g.

```
$ python main_part1.py --nsteps 10000
```

Results will be displayed after execution, and figures will be saved as .png files in the figures folder. The helpers package contains modules for Metropolis-Hasting sampling, graph generation, and sampling diagonistics, which are imported and used in the main codes. To generate pdf or html documentations of the helpers package, you need to install doxygen, and run

```
$ cd docs
$ doxygen$
```

GitHub Copilot

Copilot hasn't been shut down since it can enhance coding efficiency. However, it has been mostly used for generating repetitive codes, the overall structure and key ideas were predominantly created by myself. Codes fully generated by AI tools are all pointed out in the comments.

License

Author

Yichi Zhang (yz870)
26/03/2024

B Usage of generative AI tools

Github Copilot: Copilot hasn't been shut down since it can enhance coding efficiency. However, it has been mostly used for generating repetitive codes, the overall structure and key ideas were predominantly created by myself. Codes fully generated by AI tools are all pointed out in the comments.

ChatGPT: ChatGPT hasn't been used for any part of the code, except for some debugging and reformatting help. The report was first fully written without ChatGPT but further refinements in the delivery style of some paragraphs and the format of pseudocodes, tables and formulas were assisted by it. Instructions given to ChatGPT all fall into one of the following ways:

- Refine the following paragraph for clarity/conciseness.
- How to write a pseudocode in latex? please give an example.
- Change the following code for more readability.
- Does my understanding of something ... correct and comprehensive?
- Which comments in latex are for multiple-line formulas?
- ...

All the responses from these generative AI tools were carefully checked and fully-understood before incorporating into the code/report.