Calculating posterior probability

$$P(p|d) = \frac{P(d|p)\,P(p)}{P(d)} \quad \text{(Bayes Theorem)}$$

$P(p|d)$: Posterior probability
the probability that the model with parameter p is true, given the data d.

$P(p)$: The prior probability of parameter p, before the data was seen.

↳ In my code, I use a uniform prior so this value is constant, meaning all values of p within the specified range are equally likely.

$P(d)$: The a priori probability of witnessing data d under all possible values of p in the given model class. This is a constant that only depends on the data. This is also called the Bayesian evidence and is used to select which model class is better

↳ In my code, I calculate Bayesian evidence in parts d and e for models $y = mx$ and $y = mx + b$

$P(d|p)$: Conditional probability of seeing data d given parameters p are true. This is equal to the likelihood of the data, given the model

So then how do we calculate likelihood?

for model with input x, output y, parameter m, variance σ^2

$$\chi^2 = \sum_x \left(y_{model}(x) - y_{data}(x)\right)^2 / (\sigma^2)$$

In my problem, I wasn't given information about the variance of measurements at each x, but the error bars increased with increasing x so I assumed

$$\sigma^2 \sim x$$

In reality you could have several measurements of y_{data} at each x, so then you would have a distribution of y_{data}, the mean would be the y_{data} and the variance would be σ^2

$$\mathcal{L} = e^{-\chi^2/2}$$

If you want this likelihood $\mathcal{L}$ to represent a probability, you need to normalize it (because probabilities sum to 1)

$$\int_{m_1}^{m_2} \mathcal{L} \, dm = 1$$ (for a model with 1 parameter m)

$$\int_{b_1}^{b_2} \int_{m_1}^{m_2} \mathcal{L} \, dm \, db = 1$$ (for a model with 2 parameters m and b)

So you can use these integrals to find the normalizing factors

Part A

In the MCMC algorithm I used, we compare p_t (previous parameter value) to p' (new parameter value). I check whether p_t or p' is a better fit to the data by calculating the posterior probability for each.

Since $P(p)$ and $P(d)$ are constant, I then only have to calculate $P(d|p)$, the likelihood, in order to compare the posteriors.

$$P(p|d) \sim P(d|p) = \mathcal{L}(d|p)$$

so then

$$\frac{P(p')}{P(p_t)} = \frac{\mathcal{L}(d|p')}{\mathcal{L}(d|p_t)} > 1 \text{ if } \mathcal{L}(d|p') > \mathcal{L}(d|p_t)$$

(this means that the new parameter value is better than the previous one)

MCMC

- pick a random initial p_t from prior distribution (I picked uniform prior)
- calculate posterior probability (we actually only need likelihood for uniform prior)

  $P(p_t) \sim \mathcal{L}(p_t)$

- Pick a new p' at random. I decided I wanted my p' to be close to p_t so I picked it from a gaussian centered at p_t.

  ☆ There is an error in my code here because the way I picked p' (pnew) means that it could be outside the range of the prior I set

- Calculate $P(p') \sim \mathcal{L}(p')$
- Compare $P(p_t)$ and $P(p')$:

  $r = P(p')/P(p_t) = \mathcal{L}(p')/\mathcal{L}(p_t)$

  If $r > 1$: p' becomes the next p_t

  If $r < 1$, pick random α between 0 and 1

  ↳ If $r > \alpha$: p' becomes the next p_t

  If $r < \alpha$: stay with old p_t

This means that sometimes we will move to a worse p_t. This helps us explore more parameter space and not get stuck at a local max.

Do this loop for many iterations
(I picked t: $0 \to 10000$)

Eventually p_t will not move very much because it will be unlikely to find a p' that is better. This is when the chain has converged.

At the beginning, p_t will be dependent on the initial value chosen (p_0), so to remove this dependence we delete the first steps. (I deleted first 1000). This is called burn-in.

I plotted my chain to make sure it had converged. Graph looks like p is bouncing around some central point.

The predicted value for p (m in the 1-param case) is the value of p_t that was chosen the most.

Part B

As I was running MCMC, I kept track of how many times I accepted the new p' (in contrast with sticking with the old p_t).

The acceptance rate is the number of times I accepted the new p' divided by the number of steps (iterations).

Part C

In my likelihood function I divide by a normalizing factor, so my likelihoods represent probabilities.

Therefore, I can plot each of the pt's chosen according to their posterior probability.

The gaussian shape centered around 2 indicates that the model which had the highest probability of being true was one with $m \approx 2$.

I then printed meadian, standard deviation, and 68% confidence interval for this parameter m.

Part D

As mentioned earlier, Bayesian evidence is a measure of how well the model class fits the data. This means we are looking to see if $y = mx$ was a good model to choose or if another one (such as $y = mx + b$) could be better.

Bayesian evidence =

$$P(d|M) = \int_{m_1}^{m_2} \mathcal{L} \cdot P(m) \, dm$$

where $P(m)$ is the prior, which is a constant.

Part E

Now we find Bayesian evidence for the 2-parameter model

$$y = mx + b$$

$$P(d|M) = \int_{b_1}^{b_2} \int_{m_1}^{m_2} \mathcal{L} \cdot P(m,b) \, dm \, db$$

Where $P(m,b)$ is the priors for m and b, and is still a constant.

The Bayes factor is

$$\frac{P(d \mid y=mx+b)}{P(d \mid y=mx)}$$

to check if $y=mx+b$ is a better model than $y=mx$

Using the Jeffrey's scale I found weak evidence for $y=mx+b$ over $y=mx$.

Bonus (1)

I plotted the 2D posterior for m and b

(not using MCMC, so I have to calculate it for all combinations of m, b)

We see correlation between the posteriors of m and b which indicates a degeneracy.

Bonus (2)

The original dataset has wide error bars and the error bars are perfectly symmetric around the data points.

To add some realism I shuffle the data slightly and reduce the error bars.

I ran the same analysis as in Part A on this new data.
(I also changed the variance in the likelihood calculation by a factor of 10)

Bonus (3)

I ran MCMC for 2 parameters on the original dataset.

This meant modifying my original MCMC function for 2 parameters. The process is still basically the same

Bonus (4)

Plotted posteriors for entire parameter space for the shuffled dataset