Bayes MCMC Models

Steve Elston

10/17/2022

Review

Bayesian analysis is a contrast to frequentist methods

- The objective of Bayesian analysis is to compute a **posterior distribution**
- Contrasts with frequentist statistics with objective to compute a **point estimate** and **confidence interval** from a sample
- Bayesian models allow expressing **prior information** as a **prior distribution**
- The posterior distribution is said to quantify our current **belief**
  - *Belief is based on the posterior distribution*
  - *We update beliefs based on additional data or evidence*
  - *A critical difference with frequentist models which must be computed from a complete sample*
- Inference can be performed on the posterior distribution by finding the maximum a postiori (MAP) value and a credible interval
- Predictions are made by simulating from the posterior distribution

Review

Two functions must be defined to compute the posterior distribution

- The **likelihood** for the model being used
  - *The likelihood function includes the parameters for the model*
  - *Example; for Binomial likelihood is the probability of success*
  - *Example; for Normal likelihood is the mean, μ and variance. σ^2*
- The **prior distribution** encodes the information we have in advance about the model parameters
  - *For simple cases is the **conjugate distribution***
  - *The posterior distribution is in the conjugate family*
  - *Example, for binomial likelihood, the conjugate is the $Beta(\alpha, \beta)$*
  - *Example, for Normal likelihood, the normal distribution is the conjugate for the mean, μ*

Introduction

How can we extend Bayes models to more complex problems?

- For simple problems we can use a conjugate prior and posterior
- Unlikely posterior distribution will be a simple conjugate
- Need to perform sampling to compute approximation of complex posterior
- We need highly efficient sampling methods for complex problems

Grid Sampling Cannot Scale!

Real-world Bayes models have large numbers of parameters, into the millions

- Naive approach is simple grid sampling
  - *Sample across dimensions of the parameter space*
- Consider this thought experiment, sampling dimension 100 times:
  - *1-parameter model:* 100 *samples*
  - *2-parameter model:* $100^2 = 10000$ *samples*
  - *3-parameter model:* $100^3 = 10^5$ *samples*
  - *100-parameter model:* $100^{100} = 10^{102}$ *samples*
- Computational complexity of grid sampling has **exponential scaling** with dimensionality
- Need a better approach!

Scaling Bayesian models

How can we scale Bayesian models to 1000s of parameters?

- **Variational methods**
  - *Based on Variational calculus*
  - *A very effective and efficient method for some problems*
  - *But, proves difficult us use as a general solution*
  - *Details beyond our scope here*
- **Markov Chain Monte Carlo (MCMC)**
  - *A simple and reliable method for Bayesian inference*
  - *Our focus here*

Introduction to Markov Chain Monte Carlo

Large-scale Bayesian models need highly efficient sampling methods

- **Markov chain Monte Carlo (MCMC) sampling** is efficient and scalable
- Rather than sampling on a grid MCMC methods sample distributions efficiently
  - *Sample higher density regions of posterior with higher probability*
- Requires effort to understand how the algorithm works
  - *Must carefully evaluate how well algorithm converged*
  - *What to do when things go wrong*

What is a Markov process?

MCMC sampling uses a **Markov processes sampling chain**

- A Markov process is a **stochastic process** that transitions from a current state, x_t, to some next state, x_{t+1}, with probability Π
  - ***No dependency on past states***
- Summarize properties of a Markov process:
  - *Probability of state transition is parameterized by a matrix of probabilities, Π, of dim N X N for N possible states*
  - Π *only depends on the current state,* x_t
  - *Transition can be to current state.*

What is a Markov process?

Since Markov transition process depends only on the current state a Markov process is **memoryless**

- We can express the sequence of a Markov transition processes as:

$$p(X_{t+1}|X_t = x_t, x_{t-1}, x_{t-2}, \dots, x_0) = p(X_{t+1}|x_t)$$

- The Markov process is memoryless
  - *Transition probability only depends on the current state,* x_t
  - *No dependency on any previous states,* $\{x_{t-1}, x_{t-2}, \dots, x_0\}$.

What is a Markov process?

For system with N possible states we can write the **transition probability matrix**, Π:

$$\Pi = \begin{bmatrix} \pi_{1,1} & \pi_{1,2} & \cdots & \pi_{1,N} \\ \pi_{2,1} & \pi_{2,2} & \cdots & \pi_{2,N} \\ \cdots & \cdots & \cdots & \cdots \\ \pi_{N,i} & \pi_{N,2} & \cdots & \pi_{N,N} \end{bmatrix}$$

$$where$$

$$\pi_{i,j} = probability\ of\ transition\ from\ state\ j\ to\ state\ i$$

$$and$$

$$\pi_{i,i} = probability\ of\ staying\ in\ state\ i$$

$$further$$

$$\pi_{i,j} \neq \pi_{j,i}\ in\ general$$

Example of a Markov Process

To make the foregoing more concrete let's construct a simple example. We will start with a system of 3 states, $\{x_1, x_2, x_3\}$. The transition matrix is:

$$\Pi = \begin{bmatrix} \pi_{1,1} & \pi_{1,2} & \pi_{1,3} \\ \pi_{2,1} & \pi_{2,2} & \pi_{2,3} \\ \pi_{3,1} & \pi_{3,2} & \pi_{3,3} \end{bmatrix} = \begin{bmatrix} 0.5 & 0.0 & 0.6 \\ 0.2 & 0.3 & 0.4 \\ 0.3 & 0.7 & 0.0 \end{bmatrix}$$

- Some key points to notice
  - *The probabilities of transition from a state is given in a column*
  - *The probabilities in each column must add to 1.0*
  - *The probabilities of a transition to the same state are along the diagonal of the matrix*
- Some transitions not possible have a probability of 0.0
  - *Example,* $\pi_{2,1} = 0$*; cannot transition from state 2 to 1*
  - *Example,* $\pi_{3,3} = 0$*; cannot remain in state 3*

Example of a Markov Process

Let's apply a probability matrix to a set of possible states

- The **state vector** represents being in the first state at time step t; $\overrightarrow{x_t} = [1, 0, 0]$
- After a state transition, we compute the probability of being in each of the three possible states at the next time step, $t + 1$:

$$\vec{x}_{t+1} = \Pi \, \vec{x}_t = \begin{bmatrix} 0.5 & 0.0 & 0.6 \\ 0.2 & 0.3 & 0.4 \\ 0.3 & 0.7 & 0.0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.2 \\ 0.3 \end{bmatrix}$$

From Markov process to Markov chain

The foregoing is a single step of a Markov process

- What happens when there is a series of transitions?
- A sequence of such transitions is known as a **Markov chain**
- There are two major behaviors observed with Markov Chains
- **Episodic Markov chains** have a **terminal state**
  - *The terminal state can only transition to itself*
  - *Once the system is in the terminal state, we say that the episode has ended*
  - *Episodic processes are not of direct interest here*
- **Continuous Markov chains** have no terminal state
  - *Continue indefinitely, at least in principle*
  - *Continuous Markov chains sample probability distribution*
  - *Are ideal for estimating Bayesian posterior distributions*

From Markov process to Markov chain

Markov chain comprises a number of state transitions

- Chain of n state transitions, $\{t_1, t_2, t_3, \ldots, t_n\}$
- Each transition has the probabilities given by the state transition matrix, Π
- To estimate the probabilities of being in the states we use a special case known as a **stationary Markov chain**
  - *We will skip the technical mathematical details here*
  - *Over a large number of time steps the number of times the states are visited is proportional to the state probabilities*

From Markov process to Markov chain

Start with initial state, $\vec{x}_0$ for a continuous Markov chain:

$$\Pi \, \Pi \, \Pi \, \ldots \Pi \, \vec{x}_t = \Pi^n \, \vec{x}_t \xrightarrow[n \to \infty]{} \overrightarrow{p(x)}$$

- We can find the probabilities of the states without knowing the values of the transition matrix, Π!
- As long as we can repeatedly sample the stochastic Markov process, we can estimate the state probabilities
- This is the key to Markov Chain Monte Carlo sampling

MCMC and the Metropolis-Hastings Algorithm

The first MCMC sampling algorithm developed is the **Metropolis-Hastings (M-H) algorithm**; often referred to as Metropolis algorithm or the M-H algorithm.

- The M-H algorithm uses the following steps to estimate the posterior density:
  - *Pick a starting point in the parameter space*
  - *Sample the posterior distribution according to the model, the product of the likelihood* $P(data|parameters)$ *and prior,* $P(parameters)$
  - *Choose a nearby point in parameter space randomly and evaluate the posterior at this point*
  - *Use the following* ***decision rule to accept or reject*** *the new sample:*
    - If the likelihood, $p(data|parameters)$, of the new point is greater than the current point, accept new point
    - If the likelihood of the new point is less than your current point, only accept with probability according to the ratio:

$$Acceptance\ probability\ = \frac{p(data|new\ parameters)}{p(data|previous\ parameters)}$$

.

- If the sample is accepted, compute the posterior density at the new sample point
- Repeat sampling steps many times, until convergence

MCMC and the Metropolis-Hastings Algorithm

Eventually, the M-H algorithms converges to the posterior distribution

- M-H random sampling algorithm is far more **sample efficient** than naive grid sampling
- Consider that the M-H algorithm probabilistically samples the parameter space
  - *Preferentially sample high density areas*
  - *Not every point on a grid*
- Important properties of the Metropolis-Hastings MCMC algorithm include:
  - *The M-H algorithm is **guaranteed to eventually converge** to the underlying distribution, but convergence can be quite slow*
  - *High **serial correlation** from one sample to the next in chain gives slow convergence*

MCMC and the Metropolis-Hastings Algorithm

Poor convergence arises from low **sample efficiency**

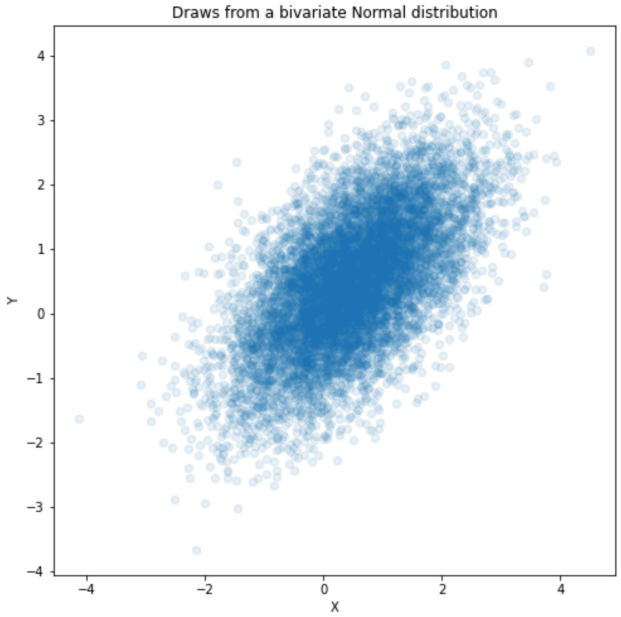
- Algorithm must be 'tuned' to ensure sample efficiency
- Tuning finds a good dispersion parameter value for the state sampling distribution
- Dispersion parameter determines the size of the jumps the algorithm makes
- Example, for Normal distribution pick the variance σ^2
  - σ^2 *is too small, the chain only slowly searches*
  - σ^2 *is too big, chain has are large jumps which slows convergence*

M-H algorithm example

Let's try a simple example, find an estimate of the posterior density of a bivariate Normal distribution

$$\mu = \begin{bmatrix} .5 \\ .5 \end{bmatrix}$$

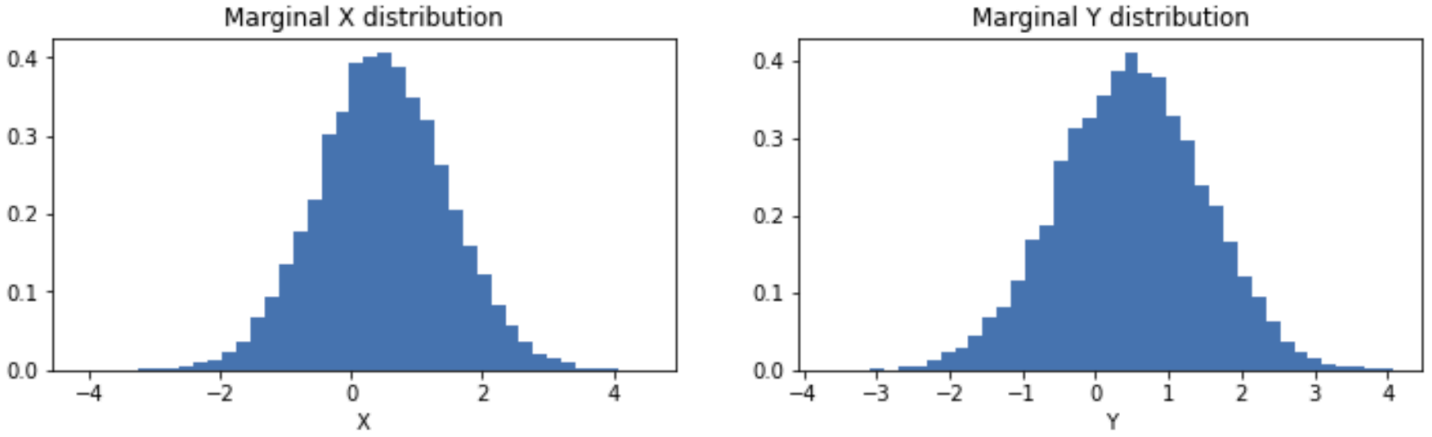
$$Covariance = \begin{bmatrix} 1, .6 \\ .6, 1 \end{bmatrix}$$



Random samples from a bivariate Normal distribution

M-H algorithm example

And, the marginal distributions of the variables



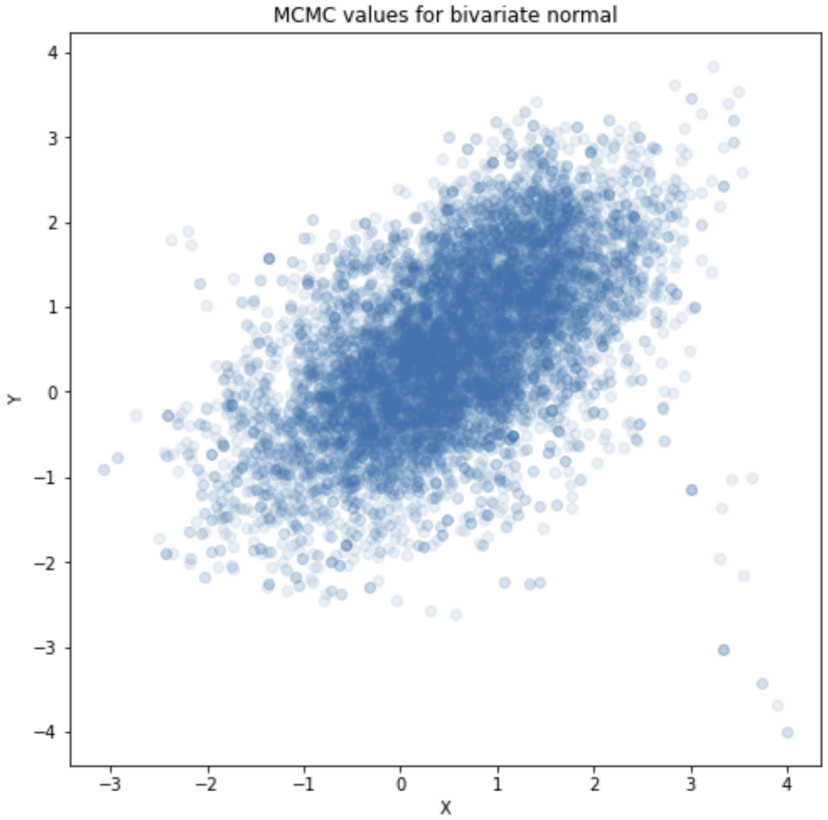
Marginal distributions of bivariate Normal samples

M-H algorithm example

Now, we are ready to sample these data using the M-H MCMC algorithm

The algorithm is

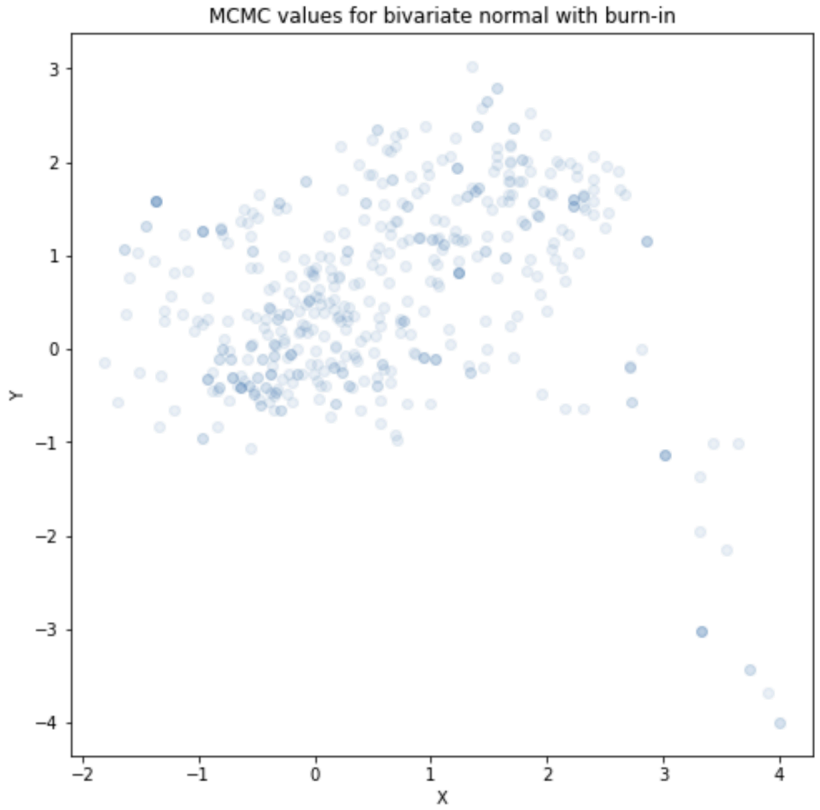
1. Compute the bi-variate Normal distribution likelihood
2. Initialize the chain
3. Initialize some hyperparameters statistics
4. Sample the likelihood of the data using the M-H algorithm.



MCMC samples from bivariate Normal distribution

M-H algorithm example

Plot the first 500 samples

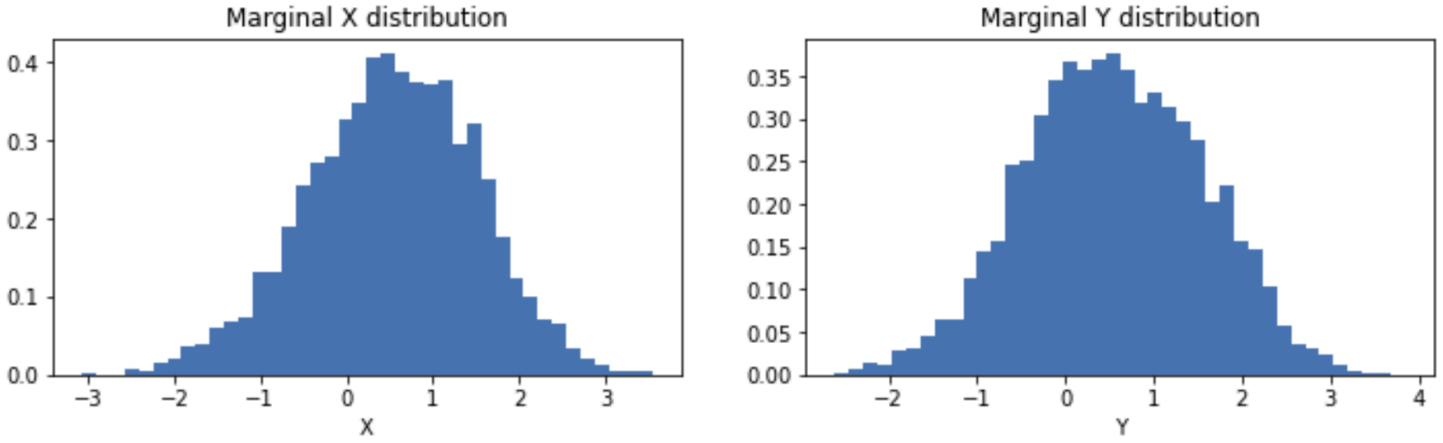


First 500 MCMC samples from bivariate Normal distribution

Notice the long 'tail' on the sampled distribution from the **burn-in period**.

M-H algorithm example

Marginal distributions of the MCMC samples, less first 500



First 500 MCMC samples from bivariate Normal distribution

These marginals are similar to the original distribution

Convergence and sampling efficiency of MCMC

How can we understand the onvergence properties of the M-H MCMC sampler

- MCMC sampling generally convergences to the underlying distribution, but can be slow
- In some pathological cases, convergence may not occur at all
- The **acceptance rate** and **rejection rate** are key convergence statistics for the M-H algorithm
  - *Low acceptnce or high rejection rate are signs of poor convergence*
  - *Too few rejections, indicate that the algorithm is not exploring the parameter space sufficiently*
  - *Trade-off between these statistics is controlled by the dispersion of the sampling distribution*
  - *Unfortunately, there are few useful rules of thumb one can use*

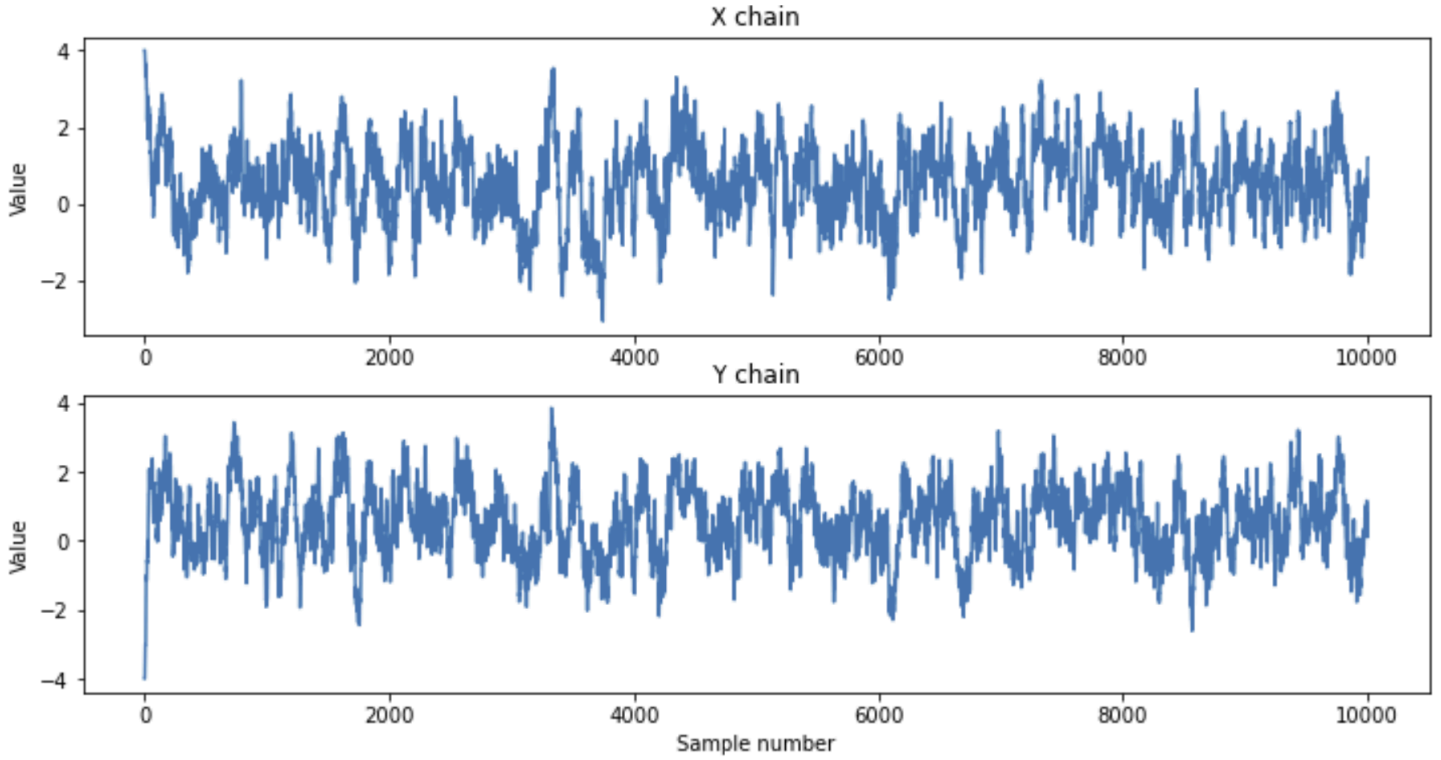
For our example these statistics are fairly good:

$$Acceptance\ rate = 0.81$$

$$Rejection\ rate = 0.19$$

Evaluation of MCMC sampling

Trace plot of the samples displays the sample value of the parameter with sample number



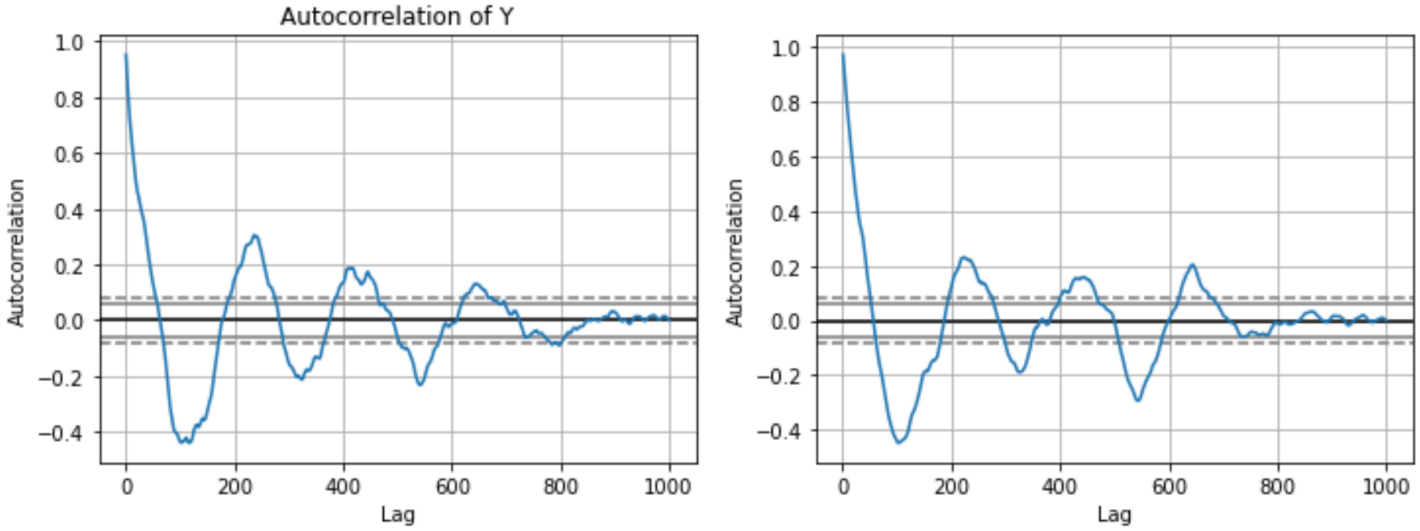
Trace plots for two variables from the M-H algorithm

The traces show sampling around the highest density values of the parameters, indicating good convergence

Evaluation of MCMC sampling

An autocorrelation plot shows how a sample value is related to the previous samples

- The autocorrelation of the Markov chain sampling



Autocorrelation of Markov chain for the two parameters

- Notice that the autocorrelation dies off fairly quickly with lag
- Low auto correlation indicates good sampling efficiency

Evaluation of MCMC sampling

We can relate sampling efficiency to the autocorrelation of the samples

- Intuitively, uncorrelated samples provide maximum information on the distribution sampled
  - *With significant autocorrelation, the new information gathered per-sample is reduced*
- **Effective sample size or ESS**.is the ratio between the number of samples adjusted for the autocorrelation and the hypothetical number of uncorrelated samples, N

$$ESS = \frac{N}{1 + 2\sum_k ACF(k)}$$

- ESS close to N indicates low autocorrelation and high sample efficiency

Other MCMC Sampling Algorithms

A number of other powerful MCMC sampling algorithms have been developed

- The M-H can suffer from slow convergence for several reasons
  - *Generally has fairly high serial correlation and low ESS*
  - *Must 'tune' the state selection probability distribution*
- As a result of these limitations, other MCMC sampling methods have been proposed in a quest to improve sample efficiency including:
  - *Gibbs sampling*
  - *No U turn sampling (NUTS)*

Gibbs sampling

Gibbs sampler is an improved MCMC sampler which speeds convergence

- Named for the 19th Century physicist Josiah Willard Gibbs; inspired by statistical mechanics
- Gibbs sampler samples each dimension of the parameter space sequentially in a round-robin manner
- M-H algorithm attempts jumps across all dimensions of the parameter space.
- Compared to the M-H, Gibbs sampling reduces serial correlation through round-robin sampling
- Update along each dimension approximately orthogonal to the preceding sampled dimension
- There are no tuning parameters since sampling is based on the marginals of the likelihood.

Gibbs sampling

The basic Gibbs sampler algorithm has the following steps:

1. For an N dimensional parameter space, $\{\theta_1, \theta_2, \ldots, \theta_N\}$, find a random starting point
2. In order, $\{1, 2, 3, \ldots, N\}$, assign the next dimension to sample, starting with dimension 1; actual order not important
3. Sample the marginal distribution of the parameter given the observations, D, and other parameter values: $p(\theta_1 | D, \theta_2, \theta_3, \ldots, \theta_N)$
4. Repeat steps 2 and 3 until convergence

Hamiltonian MCMC

The Hamiltonian sampler was proposed in 1987 (Duane, et.al.) and uses a simple idea from classical mechanics

- Attempt to improve convergence of Metropolis-Hastings algorithm
  - *M-H algorithm is slow converging* ***random walk***
  - *Leads to low ESS*
- Hamiltonian MCMC attempts to create a better directed sampling path
  - *Uses an analogy with classical physics*
  - *Constrains the search to favor high-density parts of the posterior distribution*
- Imagine that the posterior density is like a hilly landscape
  - *We want to sample around the high spots, the maximum density points*
  - *We call density* ***potential***
- We roll an imaginary ball around the landscape to the highest potential energy points
  - *Ball has position and velocity in the space*
  - *These high density points attract the ball - a 'field of attraction'*

Hamiltonian MCMC

The Hamiltonian sampler uses a simple idea from classical mechanics

- Ball has **potential energy** and **kinetic energy**
  - ***Potential energy*** *determined by probability density*
  - ***Kinetic energy*** *determined by rate of change of position,* ***velocity***
- **Hamiltonian** arise from the principle of **conservation of energy**

$$\mathrm{H}(q, p) = \mathrm{K}(p, q) + \mathrm{V}(q)$$

- Where:
  - p = *momentum vector*
  - q = *position vector*
  - $\mathrm{K}(p, q)$ = *kinetic energy*
  - $\mathrm{V}(q)$ = *potential energy*
- The Hamiltonian of the system **must remain constant** over the sample space

Hamiltonian MCMC

The Hamiltonian sampler uses a simple idea from classical mechanics

- We can relate position and momentum to the density we want to sample
- The **Boltzmann distribution** of the Hamiltonian

$$p(q, p) = e^{-\frac{\mathrm{H}(q,p)}{kT}}$$

- Where:
  - k = *normalization or* ***partition coefficient***
  - T = *temperature*

Hamiltonian MCMC

The Hamiltonian sampler uses a simple idea from classical mechanics

- **Hamiltonian** arise from the principle of **conservation of energy**

$$\mathrm{H}(q, p) = \mathrm{K}(p, q) + \mathrm{V}(q)$$

- Solve for velocity and momentum as system of coupled differential equations:

$$\frac{d\,q}{d\,t} = \frac{\partial \mathrm{H}}{\partial p} = \frac{\partial \mathrm{K}}{\partial p} + \frac{\partial \mathrm{V}}{\partial p}$$
$$\frac{d\,p}{d\,t} = \frac{\partial \mathrm{H}}{\partial q} = \frac{\partial \mathrm{K}}{\partial q} + \frac{\partial \mathrm{V}}{\partial q}$$

- Notice that HMCMC only works for distributions with finite derivatives
  - *Works for nearly all continuous distributions*
  - *Cannot be applied to discrete distributions*
- The above is rather intimidating, and we skip the details!

Hamiltonian MCMC

Hamiltonian MCMC is an extension of the M-H algorithm with steps:

1. Sample $p \sim \mathrm{N}(0, \sigma)$
2. Simulate q_t and p_t for a L time steps to time T, using the coupled differential equations
3. Get q_T as the new proposed state
4. Apply the M-H acceptance criteria to q_T

- Must find good value of L to **tune** the algorithm
  - *L too small, search is only local*
  - *L too large, search makes large jumps*
- Hard to find good hyperparameter value in practice

No U-Turn Sampler

NUTS represents the state of the art in MCMC samplers: Hoffman and Gelman 2014

- PyMC3 package uses the NUTS MCMC algorithm.
- NUTS improves on HMC
  - *Run time forward and backward when solving coupled differential equations*
  - *Find equilibrium at the* ***no U-turn point***
  - *Equilibrium point determines L automatically*
  - *Eliminates tuning*
- Why even discuss other samplers when we have the NUTS
  - *NUTS works well in many common continuous distribution cases, it is not guaranteed to converge*
  - *In some cases a Gibbs sampler works better*
  - *Use M-H for discrete distributioins*

Hamiltonian MCMC

Hamiltonian MCMC is a complex algorithm - some resources for additional details

- A Conceptual Introduction to Hamiltonian Monte Carlo, Michael Betancourt, 2017
- MCMC using Hamiltonian dynamics, Radford Niel, 2012
- Hamiltonian Monte Carlo explained, Alex Rogozhnikov, 2016

Key Steps for MCMC Bayes Modeling

Focus of modern Bayes modeling is no evalaution

1. Define the problem
2. Collect the dataset
3. Define a model, including priors
4. MCMC sample the model
5. Verify MCMC sampling convergence and sufficiency
6. Prior predictive checks - are the priors reasonable for the problem?
7. Posterior predictive checks - Do the posterior predictions agree with the observed response?

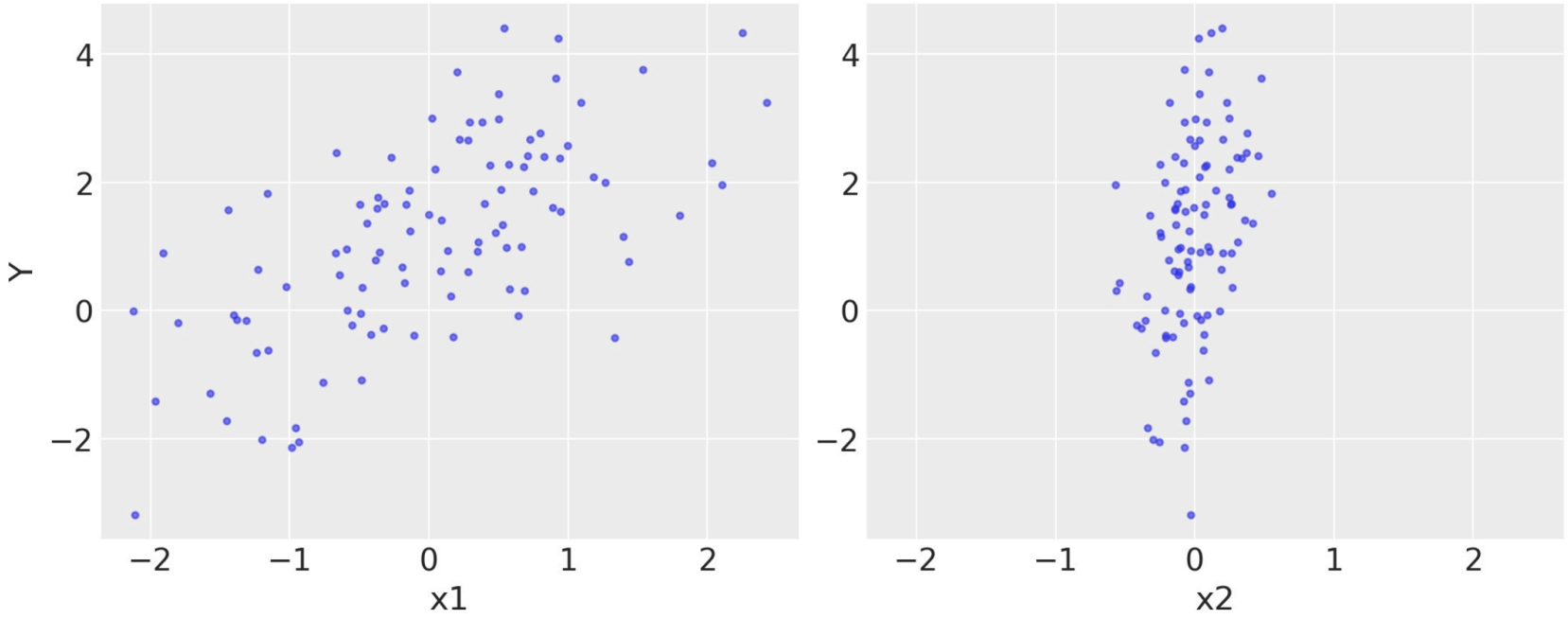
Packages like ArviZ are dedicated to evaluation of Bayes MCMC models.

Multiple examples diagnostics for the breakdown of MCMC sampling can be found in an online Appendix to a seminal paper Vehtari et.al., 2019

Constructing an MCMC Model with PyMC

Define and construct a simple linear regression model

- Simple dataset:
  - *Dependent variable* Y
  - *Independent variables* x_1, x_2



Data for regression example

Constructing an MCMC Model with PyMC

Define and construct a simple linear regression model

- Model has 4 parameters
- Define prior distributions for these parameters:

$$\begin{aligned}\beta_0 &\sim \mathrm{N}(0, 2)\\ \beta_1 &\sim \mathrm{N}(0, 2)\\ \beta_2 &\sim \mathrm{N}(0, 2)\\ \sigma &\sim |\mathrm{N}(0, 1)\end{aligned}$$

- β_0 is the intercept term
- β_1, β_2 are the partial slopes
- σ is the standard deviation
- $|\mathrm{N}(.,.)$, indicates a **half Normal distribution**

Constructing an MCMC Model with PyMC

Define and construct a simple linear regression model

- Model has 4 parameters, $[\beta_0, \beta_1, \beta_2, \sigma]$
- Likelihood model:
  - *Start with independent variables,* x_1, x_2,
  - *Sample posterior of parameters*
  - *Posterior distribution (**likelihood**) of observations,* Y*:*

$$Y \sim \mathrm{N}(\mu, \sigma)$$

- Expected value of the response, μ is **computed deterministically**:

$$\mu = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2$$

Constructing an MCMC Model with PyMC

Define and construct a simple linear regression model

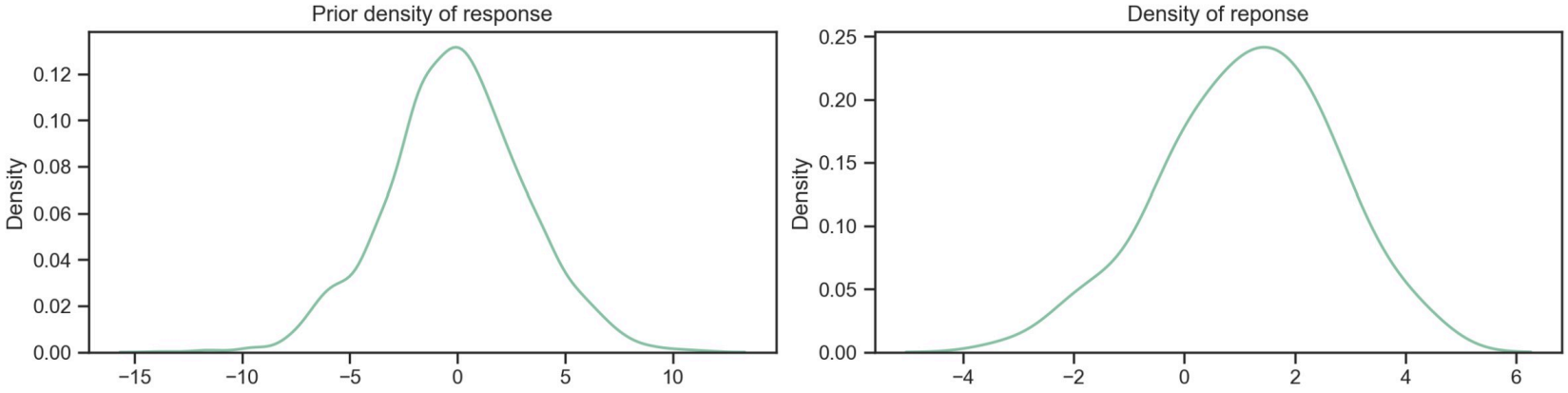
- Use Python PyMC package to construct the model

```
with pymc3.Model() as regression_model:
    # Priors for unknown model parameters
    betas = pymc3.Normal("betas", mu=0, sigma=2, shape=3)
    sigma = pymc3.HalfNormal("sigma", sigma=1)

    # Deterministic expected value of outcome
    mu = betas[0] + betas[1] * x1 + betas[2] * x2

    # Likelihood (sampling distribution) of observations
    Y_obs = pymc3.Normal("Y_obs", mu=mu, sigma=sigma, observed=Y)
```

Prior Predictive Checks

Prior predictive checks to verify prior distribution choice

- Prior predictive checks verify that choices of prior distributions are at least reasonable
- Compare the distribution of the response variable with the posterior distribution generated by priors.
  - *Is the posterior distribution in the absence of evidence (data)*



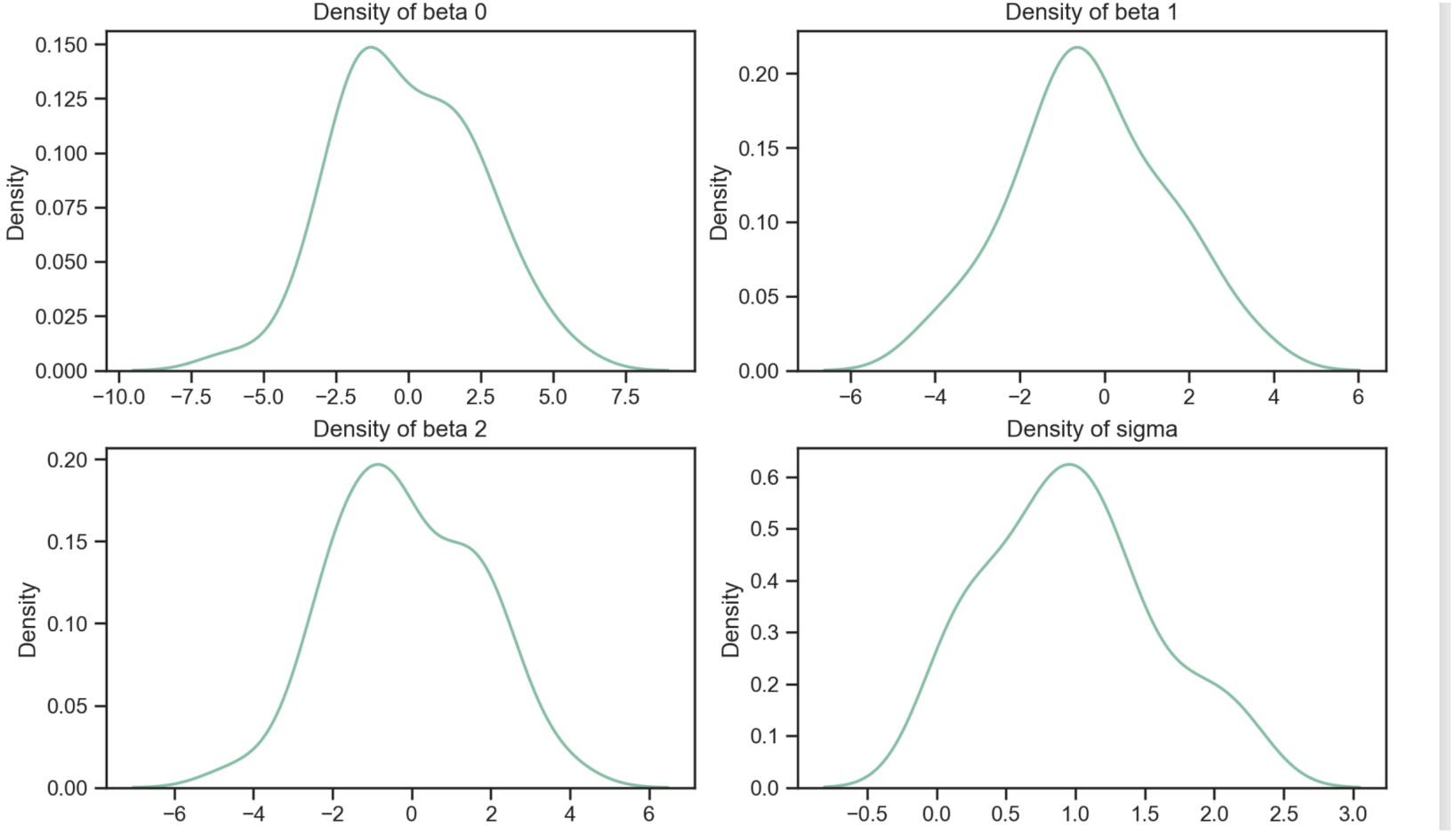
Prior predictive response vs response variable distribution

- Example looks promising
  - *Distribution shape is similar*
  - *No noticeable anomalies*
  - *Higher dispersion of prior check distribution is desired to not restrict the solution*

Prior Predictive Checks

Prior predictive checks to verify prior distribution choice

- Prior predictive checks verify that choices of prior distributions are at least reasonable
- Verify that we are happy with the density of the β and σ priors



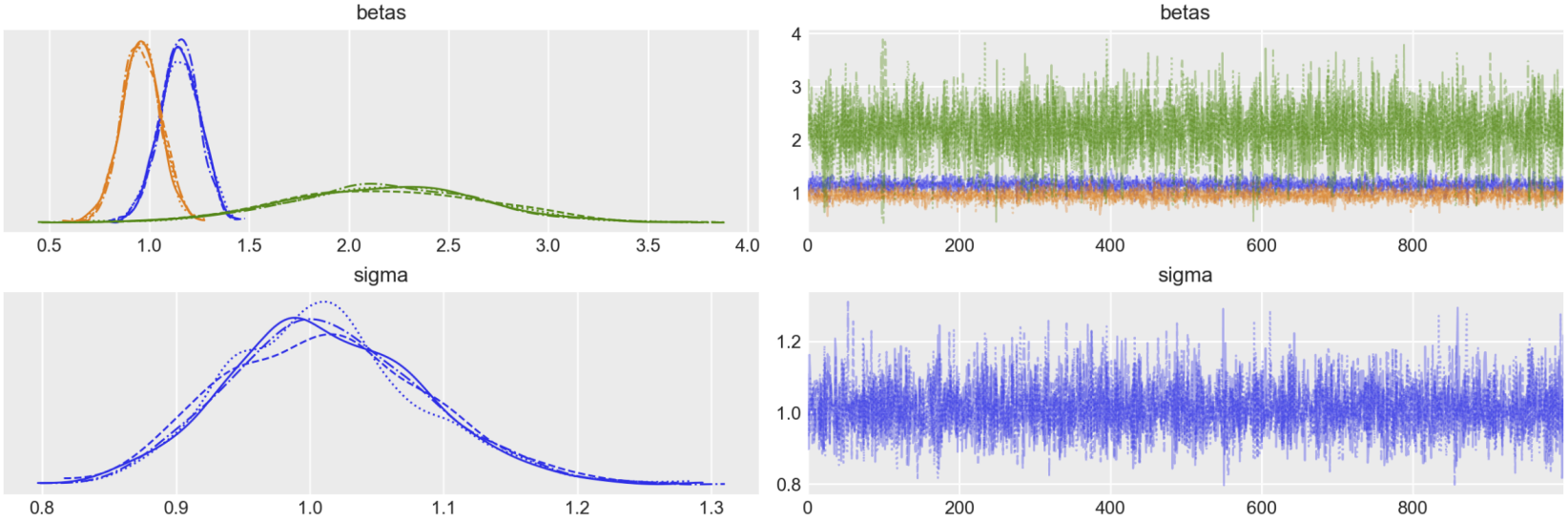
Prior density of Betas and sigma

- Shape of these distributions seems reasonable
  - *Range of possible values will not restrict the solution*
  - *No anomalies in the shape of the distribution*

Sampling and Inference with Model

Evaluation the sampled posterior and perform inference

- We sample the posterior using one or more Markov chains
- Examine the traces (path of chains)



Traces and posterior density of MCMC traces

- Traces and density of chains are similar
  - *No divergence between traces*
- 2 βs have limited uncertainty
  - *β_2 has high uncertainty as expected*
- σ has limited uncertainty

Sampling and Inference with Model

Evaluation the sampled posterior and perform inference

- There are a number of ways MCMC sampling can breakdown and not converge;

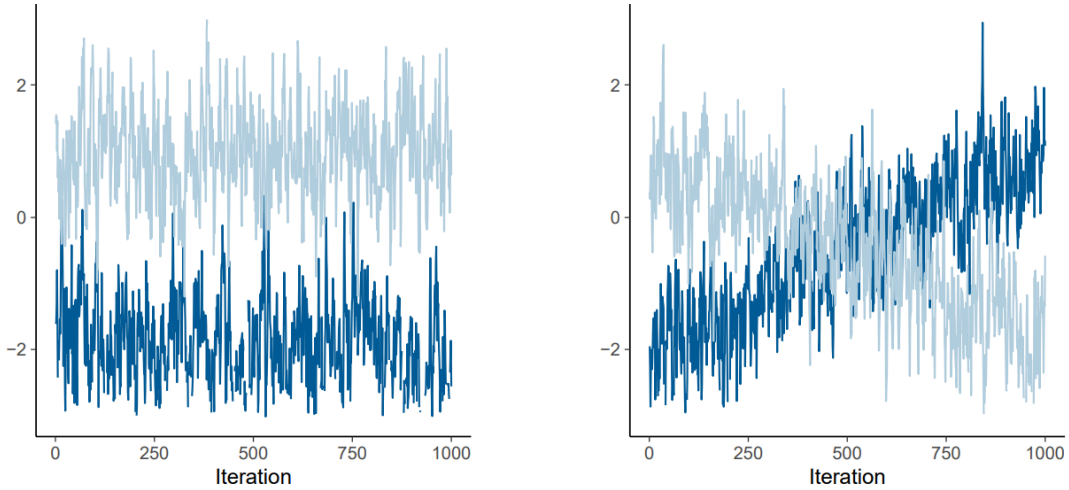


Figure 1: Examples of two challenges in assessing convergence of iterative simulations. (a) In the left plot, either sequence alone looks stable, but the juxtaposition makes it clear that they have not converged to a common distribution. (b) In the right plot, the two sequences happen to cover a common distribution but neither sequence appears stationary. These graphs demonstrate the need to use between-sequence and also within-sequence information when assessing convergence. Adapted from Gelman et al. (2013).

Example from Vehtari et.al., 2021, a seminal paper on evaluation of MCMC

Sampling and Inference with Model

We must verify that the samples are representative

	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
betas[0]	0.001	0.001	5224.0	3314.0	1.0
betas[1]	0.001	0.001	6388.0	3260.0	1.0
betas[2]	0.007	0.005	5992.0	3209.0	1.0
sigma	0.001	0.001	6987.0	2858.0	1.0

Table of MCMC convergence statistics

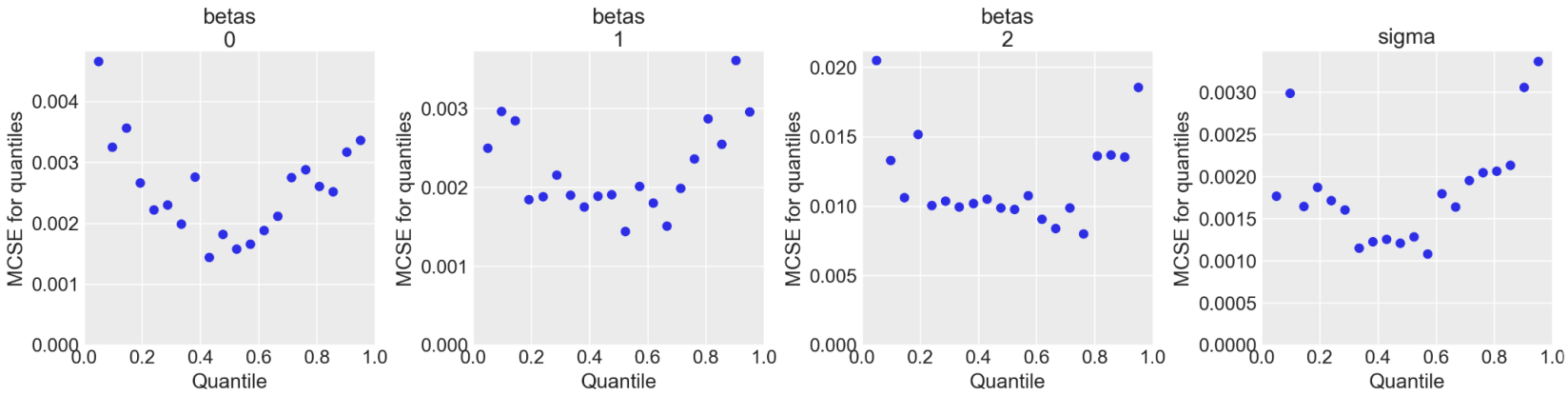
A lot of information in this summary table

1. The mean MCMC error
    - *Estimated error tells us how much error the MCMC sampling has introduced*
2. The standard deviation (`sd`) of the mean error of the posterior distribution
3. Metrics of **effective sample size (ESS)**
    - *ESS of bulk (middle) portion of posterior distributions*
    - *ESS for tails ESS of posterior distributions*
4. The **Gelman-Rudin statistic** $(\hat{R})$ measures the ratio of the **variance shrinkage between chains** to the **variance shrinkage within chains**
    - *Gelman-Rudin statistic should converge to 1.0*
    - *If all chains converge, the reduction in variance between chains and within the chains should is same*

Sampling and Inference with Model

We must verify that the samples are representative

- Plot the MCSE by quantile for each model parameter



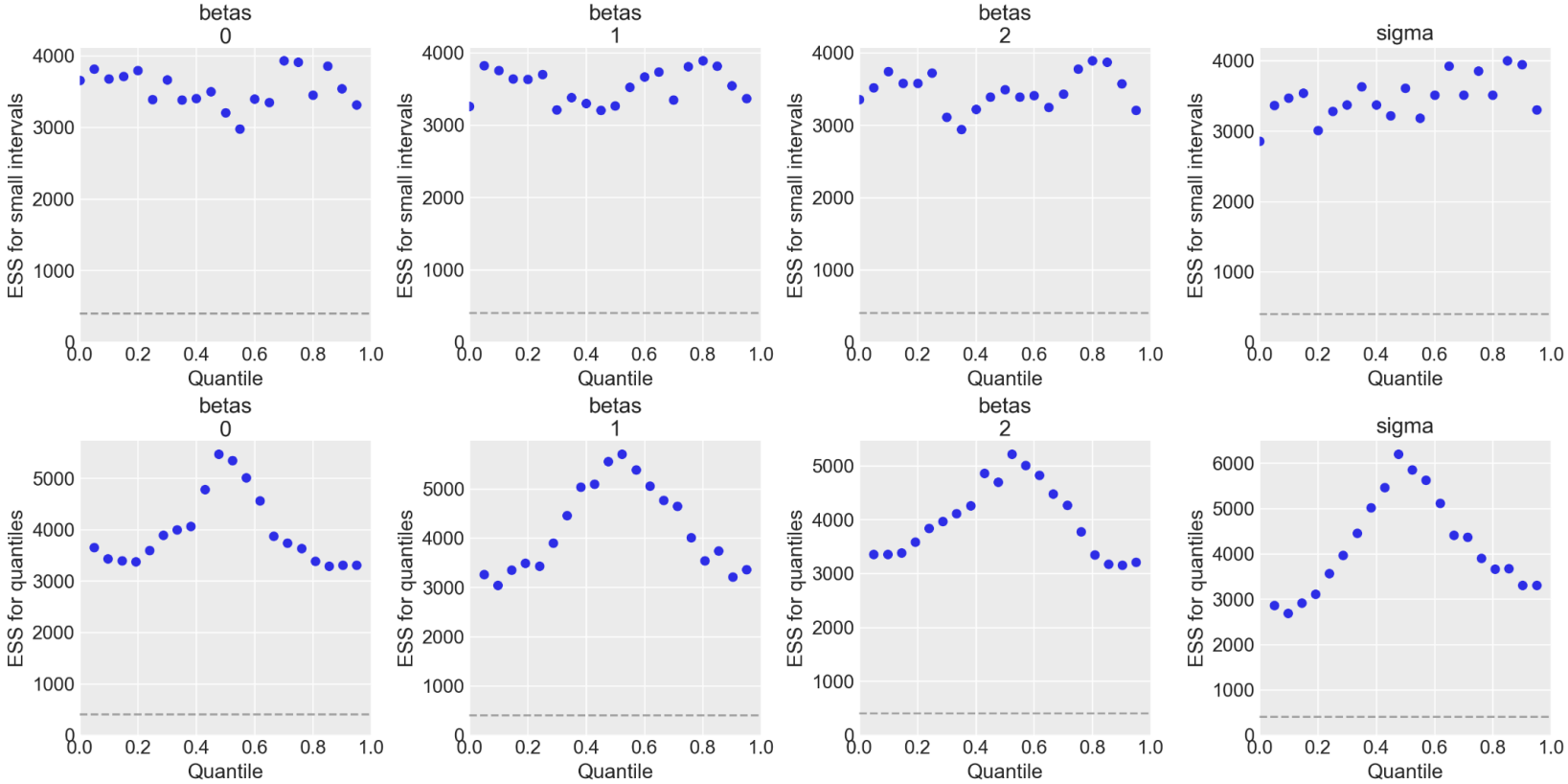
MCSE by quantile for each model parameter

- Ideally want MCSE to be **small and uniform** with quantile
- Notice the MCSE is higher for outer quantiles
  - *Not an unusual situation since less sampling in thin tails of distributions*
  - *Overall MCSE is low, even in tails*

Sampling and Inference with Model

We must verify that the samples are representative

- Plot the ESS locally and by quantile for each model parameter



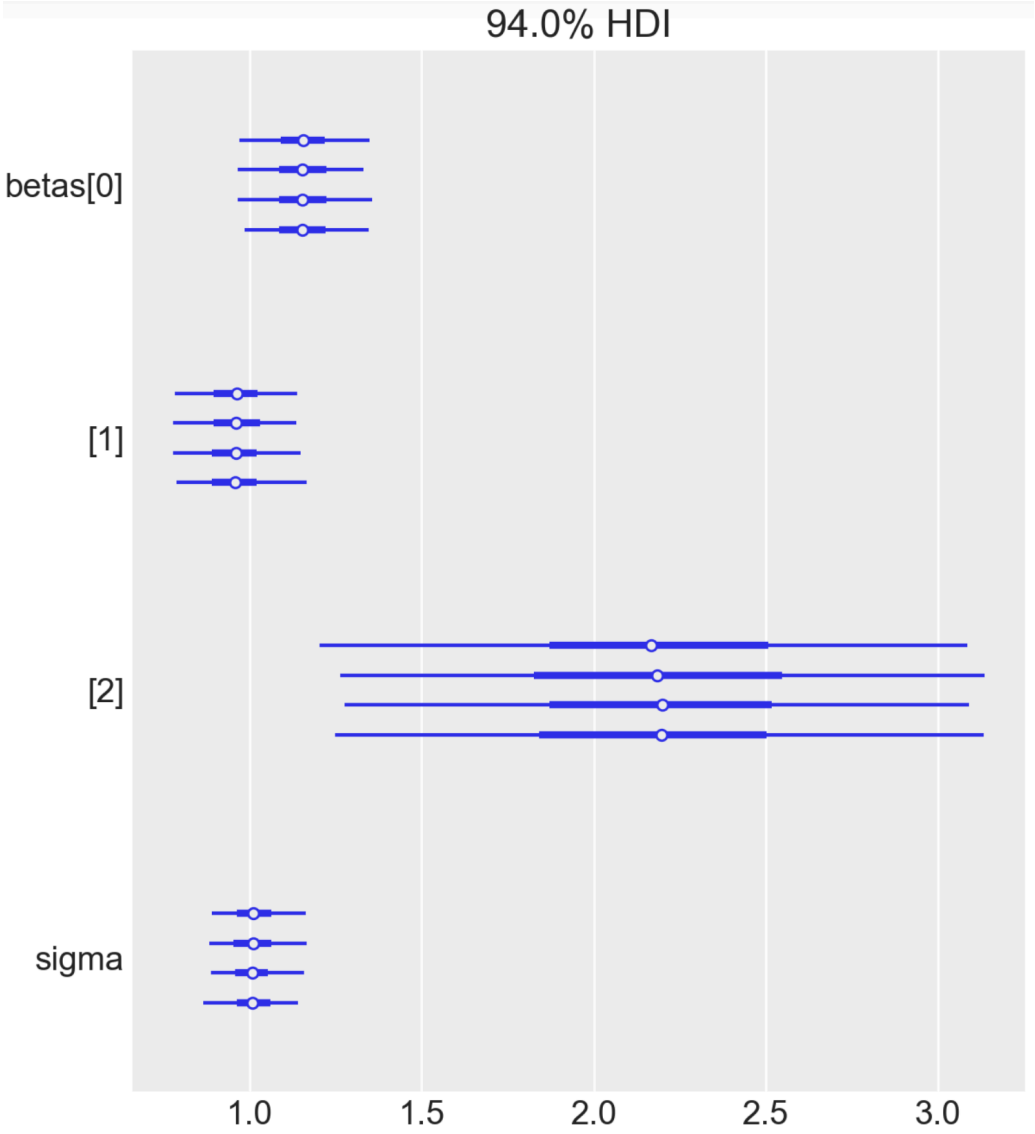
ESS locally and by quantile for each model parameter

- Ideally want ESS to be **large and uniform** with quantile
- Local ESS is fairly uniform with quantile
- Notice the quantile ESS is lower for outer quantiles
  - *Not an unusual situation since less sampling in thin tails of distributions*
  - *Overall ESS is reasonably high, even in tails*

Sampling and Inference with Model

Evaluate the sampled posterior and perform inference

- Examine HDIs of model parameters by trace with **tree plot**



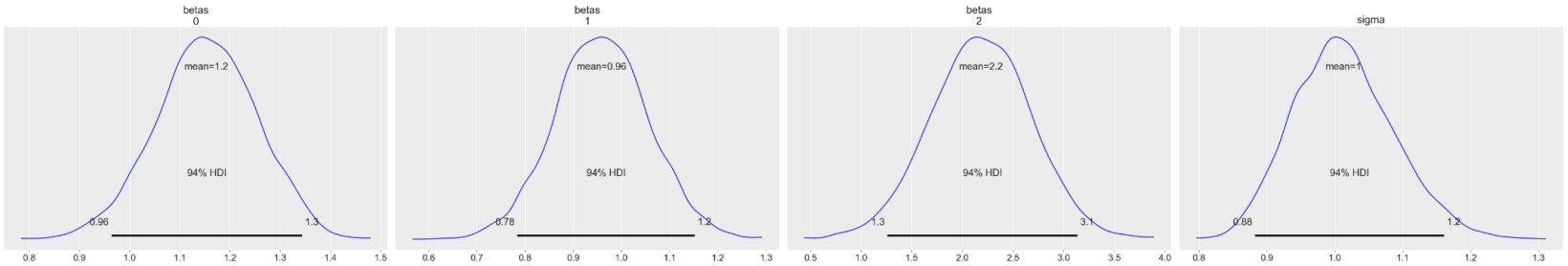
HDIs of model parameters by trace

- HDIs are similar for each parameter by trace
- HDIs confirm inferences on uncertainty of the model parameters

Sampling and Inference with Model

Evaluate the sampled posterior and perform inference

- How can we interpret the results?



Posterior distribution of parameters with HDIs

- Interpret βs by examining HDIs
  - β_0 *is the mean response for the centered independent variable*
  - β_1 *is partial slope with respect to* x_1
  - β_2 *coefficient large compared to* β_1 *with high uncertainty*
- σ has small value with limited uncertainty

Posterior Predictive Checks

We must check that predictions from the model are reasonable

- Does the posterior distribution look like the distribution of the observed response?
- Can simply compare density plots
- Apply a test statistic T
  - *Bayesian p-value*
  - *Bayesian u-value*
  - *Others*

Posterior Predictive Checks

We must check that predictions from the model are reasonable

- Base statistic on **posterior predictive distribution**
- Approximate PPD by resampling from the posterior distribution:

$$p(y^{rep}|y) = \sum_i p(y^{rep}|\theta)p(\theta|y)$$

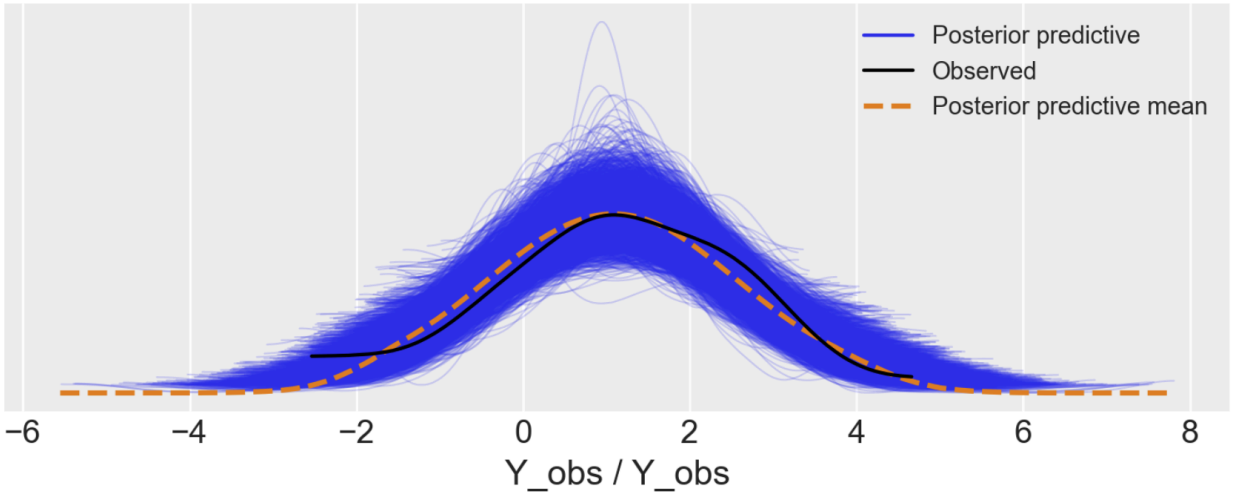
Where:
y^{rep} = realization drawn from the posterior distribution
y = observation
$p(y^{rep}|\theta)$ = draw from posterior distribution with parameters θ
$p(\theta|y)$ = posterior distribution of model parameters, θ

- We can sample the posterior as many times as required to get better estimates of $p(y^{rep}|y)$

Posterior Predictive Checks

We must check that predictions from the model are reasonable

- Posterior predictive distribution is compared to the distribution of observations



Posterior predictive distribution vs. observed responses

- Agreement is generally good between posterior predictive and actual observations
- Average predictive is heavier in the tails

Posterior Predictive Checks

We must check that predictions from the model are reasonable

- **Bayesian p-value** is a test statistic, T, for the distribution differences between observed and predicted responses:

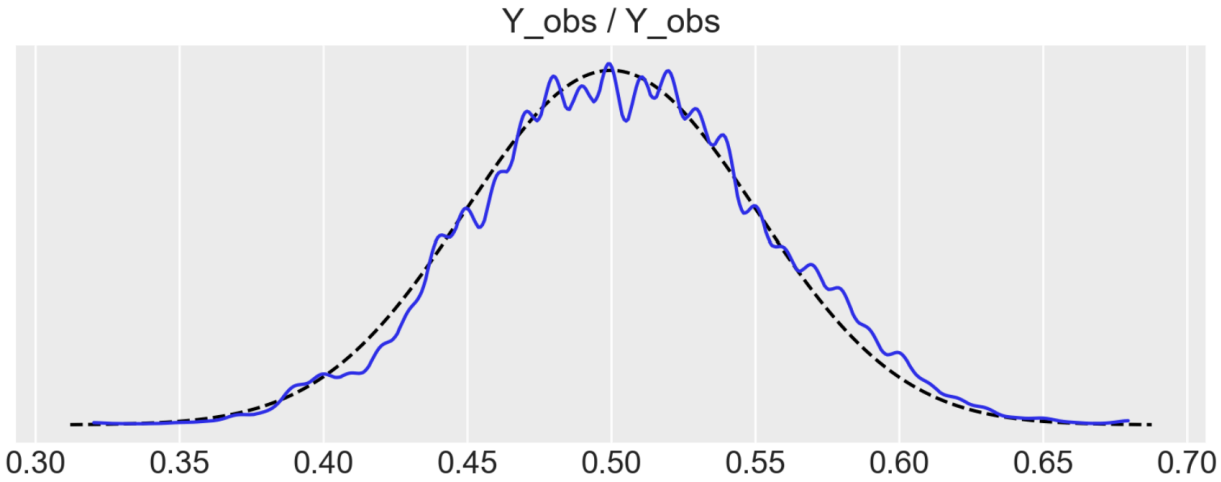
$$p = p(y^{rep} < y|y)$$

- Intuitively, for representative posterior, $p = 0.5$ over many draws of y^{rep}
  - $y^{rep} < y|y$ *half the time*
  - $y^{rep} > y|y$ *half the time*
  - *Distribution should be symmetric*

Posterior Predictive Checks

We must check that predictions from the model are reasonable

- Posterior predictive Bayesian p-value



Distribution of Bayesian p-values

- Plot is centered on 0.5
- Distribution is symmetric

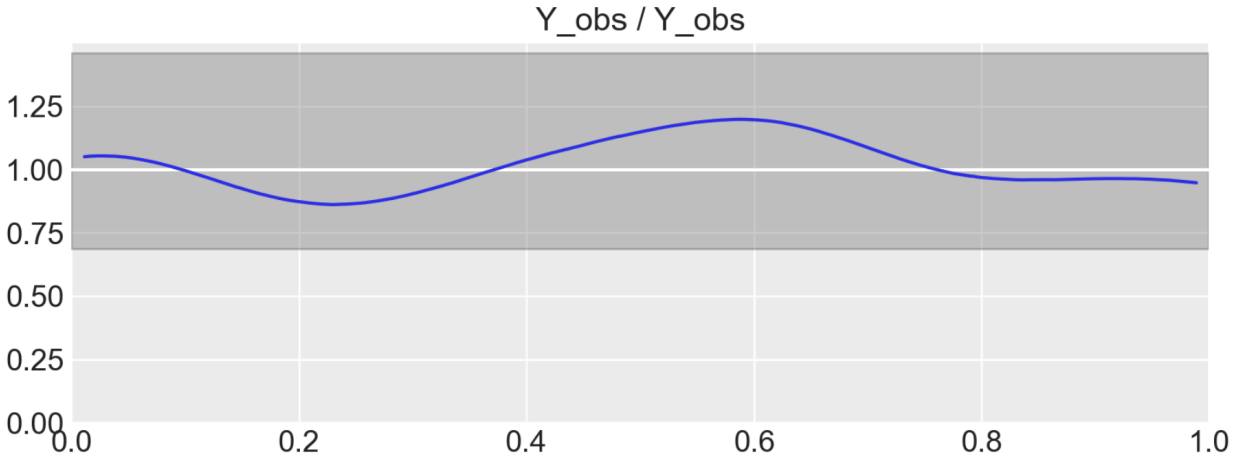
Posterior Predictive Checks

We must check that predictions from the model are reasonable

- Posterior predictive **Bayesian u-value** is also known as the **marginal p-value**:

$$p_i = p(y_i^{rep} < y_i | y)$$

- U-value computed for specific (ordered) observation y_i
  - p_i *should be centered on 1.0*
  - p_i *should be close to uniformly distributed*



Distribution of Bayesian u-values

- The values are close to 1.0 across all values of the observatons

British Coal Mine Disasters

Analysis of change point in British coal mine disasters

- Toward the end of 19th Century was growing public awareness of the rising coal mine deaths
- Mine disaster defined as incident leading to 6 or more deaths
- As early as 1886, Royal Commission published recommendations for safety practices
  - *Adoption of these practices remained voluntary*
  - *British Coal Mines Act of 1911 finally codified mine safety procedures into law.*

British Coal Mine Disasters

Analysis of change point in British coal mine disasters

- Famous dataset from Kaggle, and other sources
- Goal to create a model to quantify the differences in the rate of mine disasters before and after the introduction of safety practices
- Example of a **point process model**
- Point process model has an **intensity** or **rate** of occurrence of events, e.g. mine disasters
- Number events over a time period for this type of point process is iid
  - *The number of events per time period are a iid draw from some probability distribution*
  - *Because of the iid nature of the point process, no autoregressive (AR) effect*

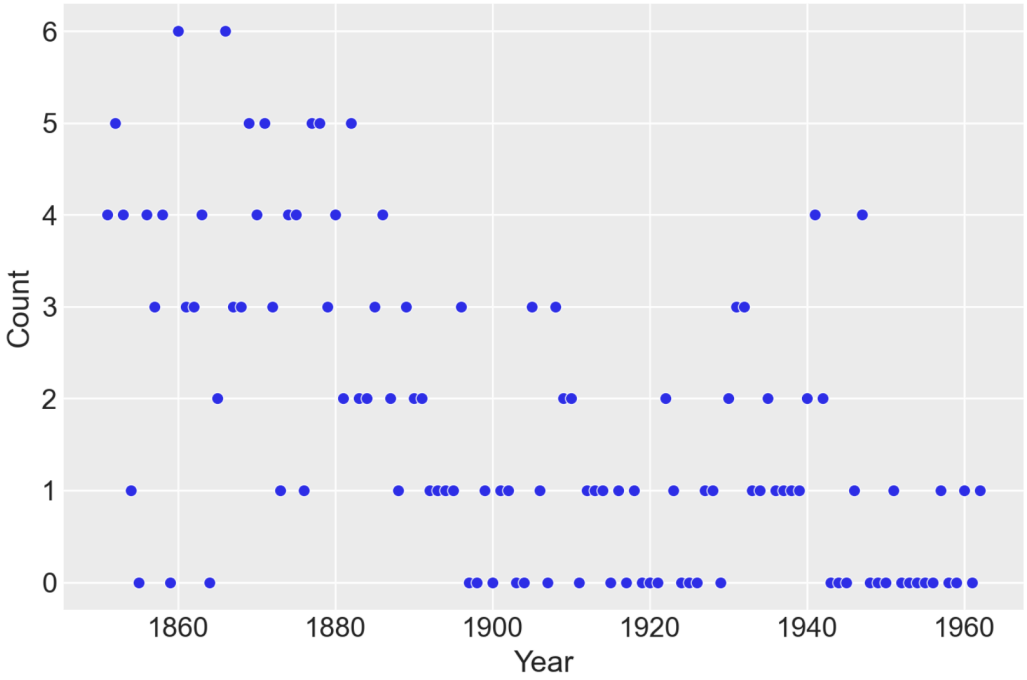
British Coal Mine Disasters

Analysis of change point in British coal mine disasters

- iid nature of the point process does not mean the distribution of the process cannot change in time
- **Change points** or **switch points** in time are quite common in the real-world
  - *Rates changes at change point*
  - *Rate constant betwen change point*
  - *Identifying these change points is often a challenging problem*

British Coal Mine Disasters

Analysis of change point in British coal mine disasters



Number of disasters per year 1851-1962

- Two related questions we can ask about these data
  - *At what point in time was there a significant change in intensity, the switch point?*
  - *Not obvious, changes in safety practice were adopted over about 25 years*
  - *How significant was the reduction in the rate?*

British Coal Mine Disasters

Analysis of change point in British coal mine disasters

- **Poisson likelihood** model with switch switch point s, rate before, e and after l:

$$D_t \sim Pois(r_t),\ r_t \begin{cases} e, & t < s \\ l, & t \geq s \end{cases}$$

- Prior distributions of parameters, uniform and exponential:

$$\begin{aligned} s &\sim unif(t_{min}, t_{max}) \\ e &\sim exp(2) \\ l &\sim exp(2) \end{aligned}$$

British Coal Mine Disasters

Analysis of change point in British coal mine disasters

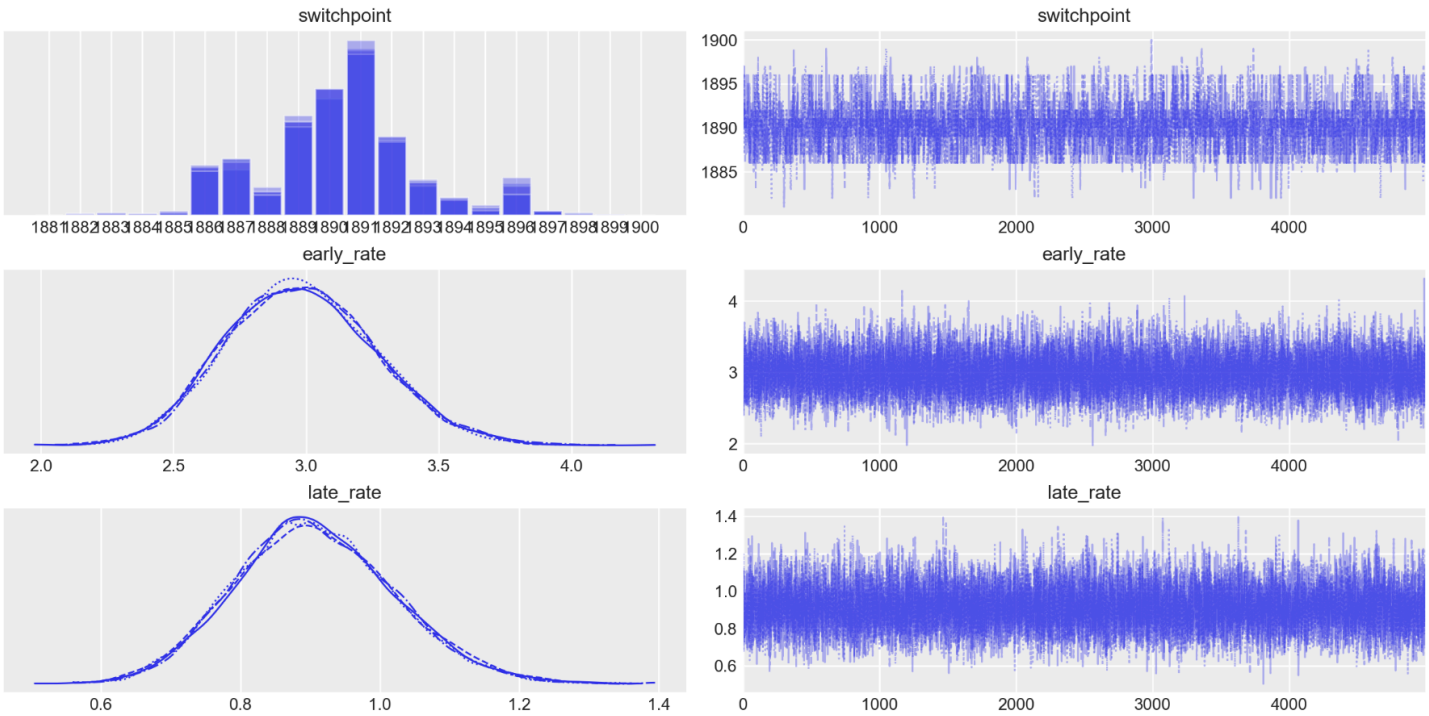
- Define model using PyMC

```
with pymc3.Model() as disaster_model:
    # Uniform prior on the switch point
    switchpoint = pymc3.DiscreteUniform("switchpoint", lower=disaster_data.Year.min(), upper=disaster_data.Year.max())
    # More informative prior based on domain knowledge
#    switchpoint = pymc3.Triangular("switchpoint", lower=disaster_data.Year.min(), upper=disaster_data.Year.max(), c=1900)

    switchpoint = pymc3.DiscreteUniform("switchpoint", lower=disaster_data.Year.min(), upper=disaster_data.Year.max())

    # Priors for pre- and post-switch rates
    early_rate = pymc3.Exponential("early_rate", 2.0)
    late_rate = pymc3.Exponential("late_rate", 2.0)

    # Poisson rate switch for years before and after current
    rate = pymc3.math.switch(switchpoint >= disaster_data.Year, early_rate, late_rate)

    disasters = pymc3.Poisson("disasters", rate, observed=disaster_data.Count)
```

British Coal Mine Disasters

Analysis of change point in British coal mine disasters

- Distributions and traces from sampling



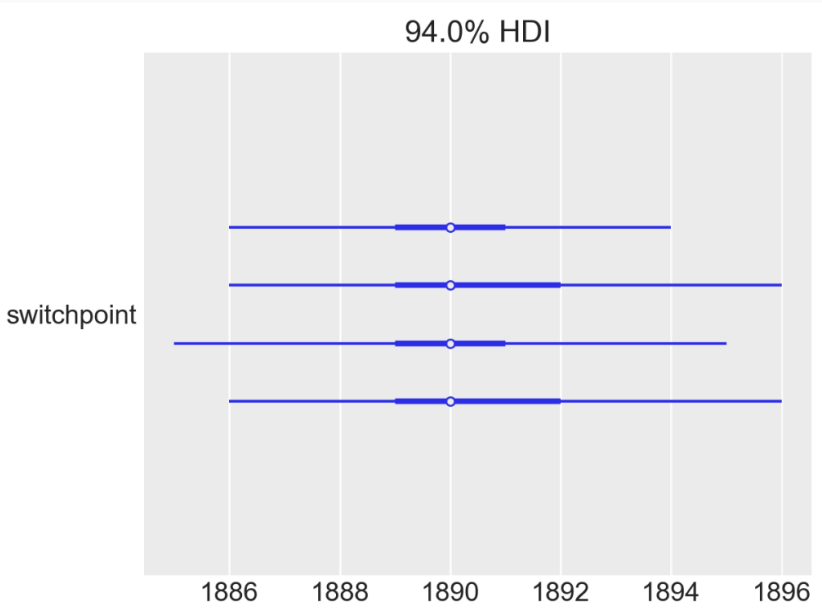
HDI of parameters by trace

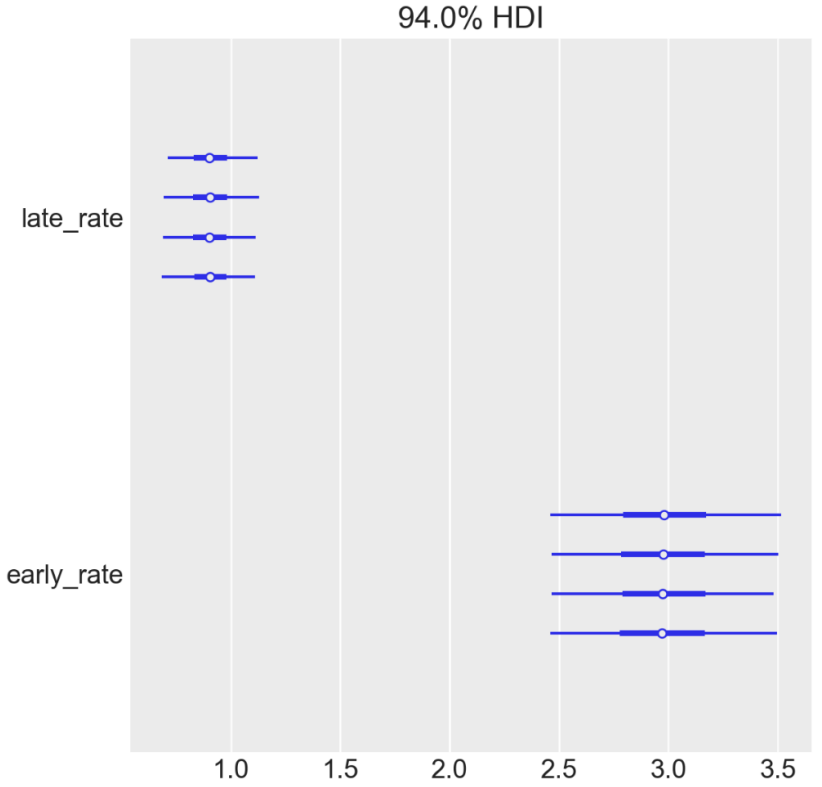
- Good agreement between traces

British Coal Mine Disasters

Analysis of change point in British coal mine disasters

- Examine the posterior HDI of the parameters for each trace





HDI of parameters by trace

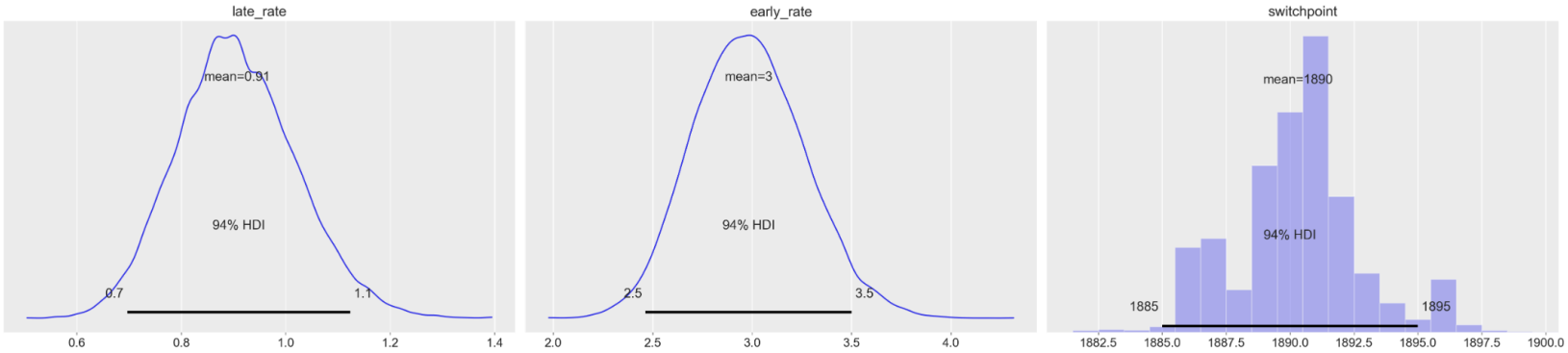
- The HDIs are consistent across traces

- See notebook for MCMC, Prior and Posterior checks

British Coal Mine Disasters

Analysis of change point in British coal mine disasters

- Examine the posterior HDI of the parameters



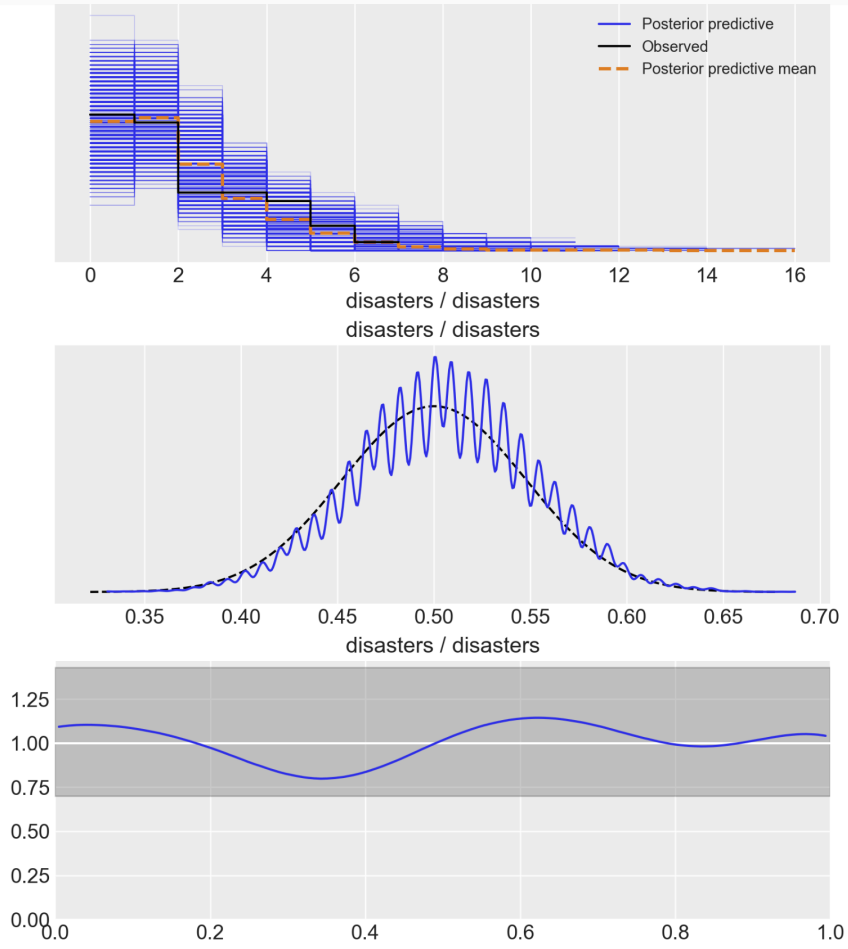
Posterior distribution of parameters with HDI

- HDI of switch point is in reasonable range
- Significant difference in HDI of rates

British Coal Mine Disasters

Analysis of change point in British coal mine disasters

- Posterior Predictive checks



Posterior predictive checks

Summary

For complex Bayesian models we need a computationally efficient aproximation

- Grid sampling is inefficient
- MCMC sampling is based on Markov chains
  - *Markov process is memoryless*
  - *Sampling converges to probability distribution*
- Several MCMC sampling methods have been developed
  - *Metropolis-Hastings (M-H) algorithm uses random sampling with acceptance probability*
  - *Gibbs sampling round robins on the dimensions of the parameter space*
  - *NUTS*
- NUTS is the state of the art MCMC algorithm
  - *Uses a field of attraction to the highest density parts of the parameter space*
  - *No hyperparameters to tune*

Summary

Performance metrics of MCMC sampling

- Sample efficiency
  - *Serial correlation reduces sample efficiency*
  - *ESS*
- Convergence
  - *Multiple chains should converge to the similar distributions*
  - *Want between chain and within chain variance to be the same*