# Lecture 5 - Computational Bayesian Statistics

## Computational Statistics and Applications

Vu Quoc Hoang (vqhoang@fit.hcmus.edu.vn)
Tran Thi Thao Nhi (thaonhitt2005@gmail.com)
FIT - HCMUS

February 5, 2023

# Agenda

1. Bayesian inference
2. Binomial probability
3. “Deriving” posterior distribution
4. Markov chain Monte Carlo
5. Probabilistic programming
6. Generalized linear model

# Agenda

1. **Bayesian inference**
2. Binomial probability
3. "Deriving" posterior distribution
4. Markov chain Monte Carlo
5. Probabilistic programming
6. Generalized linear model

# Bayes' rule

On sample space $\Omega$, events $\{E_1, E_2, ..., E_n\}$ form a partition if

- $E_i \cap E_j = \varnothing$ as $i \neq j$ (mutually exclusive),
- $\Omega = \bigcup_{i=1}^{n} E_i$ (exhaust all possibilities).

**Bayes' rule**

$$\underbrace{P(E_i|D)}_{\text{posterior}} = \frac{\overbrace{P(D|E_i)}^{\text{likelihood}}\overbrace{P(E_i)}^{\text{prior}}}{\underbrace{P(D)}_{\text{evidence}}} = \frac{P(D|E_i)P(E_i)}{\sum_{j=1}^{n} P(D|E_j)P(E_j)} \propto P(D|E_i)P(E_i).$$

Bayes' rule shows how to "update probabilities" or, equivalently, "reallocate **credibilities**".

# Bayesian inference

**Data** is formed by a **probabilistic model**, which is specified by the various **parameter** values. When we observe data, we **infer** the model from its parameters. In general, **Bayesian inference** follows these steps:

1. Identify the data, a descriptive model, and an appropriate mathematical form for its parameters (i.e. likelihood function),
2. "Specify" the prior distribution on the parameters,
3. "Derive" the posterior distribution of parameter values from the Bayes' rule

$$\underbrace{p(\theta|D)}_{\text{posterior}} = \frac{p(D|\theta)p(\theta)}{p(D)} = \frac{\overbrace{L(\theta|D)}^{\text{likelihood}}\overbrace{p(\theta)}^{\text{prior}}}{\underbrace{p(D)}_{\text{evidence}}} \propto L(\theta|D)p(\theta),$$

4. Check that the posterior predictions mimic the data with reasonable accuracy (i.e., "posterior predictive check"). If not, then consider a different descriptive model,
5. "Use" the posterior distribution to conclude.

# Bayesian inference - Example 1

**Problem.** Suppose that there is a manufacturer of inflated bouncy balls, which are produced in 4 discrete sizes 1, 2, 3, 4. Suppose we submit an order to the factory for 3 balls of size 2. After receiving them, we find that the 3 balls have diameters of 1.77, 2.23, and 2.70. Can we conclude that the factory produce the correct type we ordered?

**Inference.** We specify a mathematical form for the ball diameters, which is random variable $X \sim \mathcal{N}(\mu, \sigma^2)$ where the parameter $\mu$ takes one of four values below

$$\begin{cases} E_1 : \mu = \mu_1 = 1.0, \\ E_2 : \mu = \mu_2 = 2.0, \\ E_3 : \mu = \mu_3 = 3.0, \\ E_4 : \mu = \mu_4 = 4.0. \end{cases}$$

We have

$$p(X = x|E_i) = f_{\mathcal{N}(\mu_i, \sigma^2)}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu_i)^2}{2\sigma^2}}, i = 1, ..., 4.$$

# Bayesian inference - Example 1 (cont.)

Hence we obtain that $X_1, X_2, X_3$ are independent and identically distributed copies of $X$. In particular, we have

$$D = (X_1 = 1.77) \cap (X_2 = 2.23) \cap (X_3 = 2.70).$$

The likelihood function for $\mu$ on the data $D$ according to the chosen model is

$$\begin{aligned} L(\mu_i|D) &= p(D|E_i) = p\left((X_1 = 1.77) \cap (X_2 = 2.23) \cap (X_3 = 2.70)|E_i\right) \\ &= p(X_1 = 1.77|E_i)p(X_2 = 2.23|E_i)p(X_3 = 2.70|E_i) \\ &= \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^3 e^{\frac{-\left((1.77-\mu_i)^2+(2.23-\mu_i)^2+(2.70-\mu_i)^2\right)}{2\sigma^2}} \\ &\propto e^{\frac{-\left((1.77-\mu_i)^2+(2.23-\mu_i)^2+(2.70-\mu_i)^2\right)}{2\sigma^2}}, i = 1, ..., 4. \end{aligned}$$

"Assume" that the prior distribution is given by

$$p(E_i) = \frac{1}{4} \propto 1, i = 1, ..., 4.$$

# Bayesian inference - Example 1 (cont.)

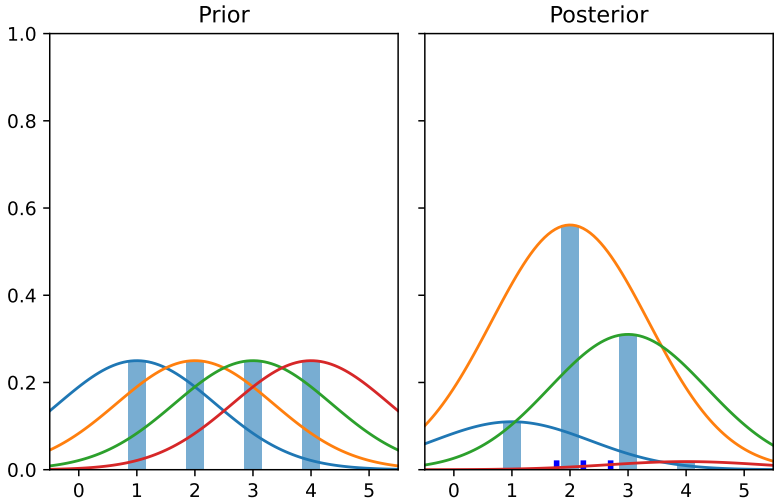
Using Bayes' rule, the posterior distribution is defined as

$$p(E_i|D) = \frac{p(D|E_i)p(E_i)}{\sum_{j=1}^{4} p(E_j|D)p(E_j)} = \frac{e^{\frac{-\left((1.77-\mu_i)^2+(2.23-\mu_i)^2+(2.70-\mu_i)^2\right)}{2\sigma^2}}}{\sum_{j=1}^{4} e^{\frac{-\left((1.77-\mu_j)^2+(2.23-\mu_j)^2+(2.70-\mu_j)^2\right)}{2\sigma^2}}}, i = 1, ..., 4.$$

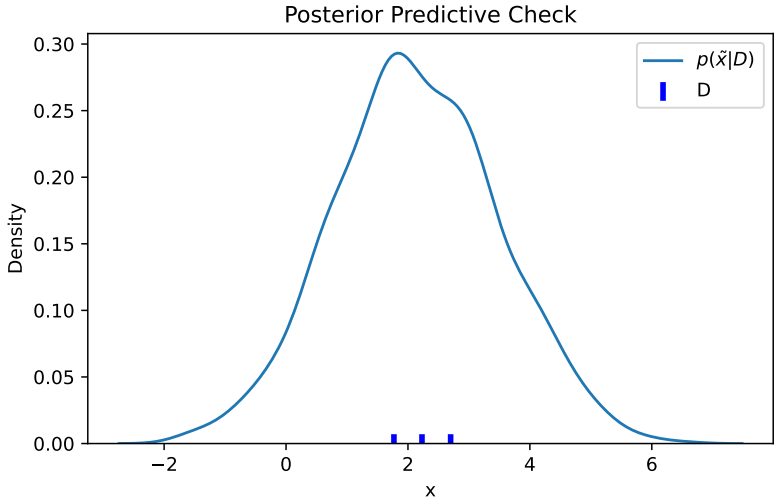
As some special cases for $\sigma^2$, we compute

| | $\sigma^2 = 1$ | $\sigma^2 = 1.35$ | $\sigma^2 = 2$ |
|---|---|---|---|
| $P(\mu = 1\|D)$ | 7% | 11% | 16% |
| $P(\mu = 2\|D)$ | 64% | 56% | 47% |
| $P(\mu = 3\|D)$ | 29% | 31% | 32% |
| $P(\mu = 4\|D)$ | 1% | 2% | 5% |

# Bayesian inference - Example 1 (cont.)



# Bayesian inference - Example 1 (cont.)



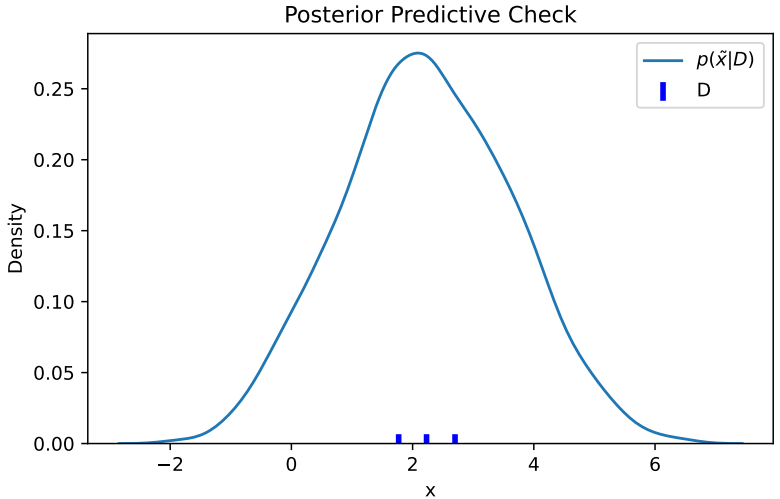
# Bayesian inference - Example 1 - PyMC3

```
import pymc3 as pm
import theano

mu = np.array([1.0, 2.0, 3.0, 4.0])
prior = 1/4 * np.ones(len(mu))
x = np.array([1.77, 2.23, 2.70])
sigma2 = 1.35

with pm.Model() as model:
    mu_index_var = pm.Categorical("mu_index", p=prior)
    mu_var = theano.shared(mu)[mu_index_var]
    x_var = pm.Normal("x", mu=mu_var, tau=1/sigma2,
       observed=x)
    trace = pm.sample(1000, return_inferencedata=False)
```

# Bayesian inference - Example 1 - PyMC3 (cont.)

```
mu_index_posterior = trace["mu_index"]
values, counts = np.unique(mu_index_posterior,
   return_counts=True)
print(mu[values])
# [1. 2. 3. 4.]
print(counts/np.sum(counts))
# [0.1225 0.5425 0.3135 0.0215]

x_post_sample = pm.sample_posterior_predictive(trace, 1000,
    model)["x"].flatten()
sns.kdeplot(x_post_sample)
plt.scatter(x, np.zeros(len(x)))
```

# Bayesian inference - Example 1 - PyMC3 (cont.)

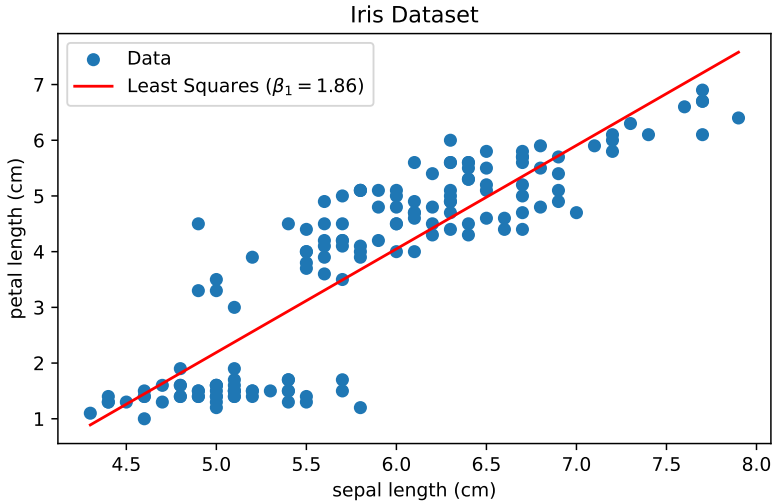


# Bayesian inference - Example 2

Iris Dataset

- Wikipedia: `https://en.wikipedia.org/wiki/Iris_flower_data_set`
- UCI: `https://archive.ics.uci.edu/ml/datasets/iris`
- Scikit-learn: `https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html`

# Bayesian inference - Example 2 (cont.)



# Bayesian inference - Example 2 (cont.)

**Problem.** From Iris dataset, determine the dependence of petal-length on sepal-length.

**Inference.** "Based on the scatter plotted data", we form a dependence model of petal-length ($y$) on sepal-length ($x$) as follows:

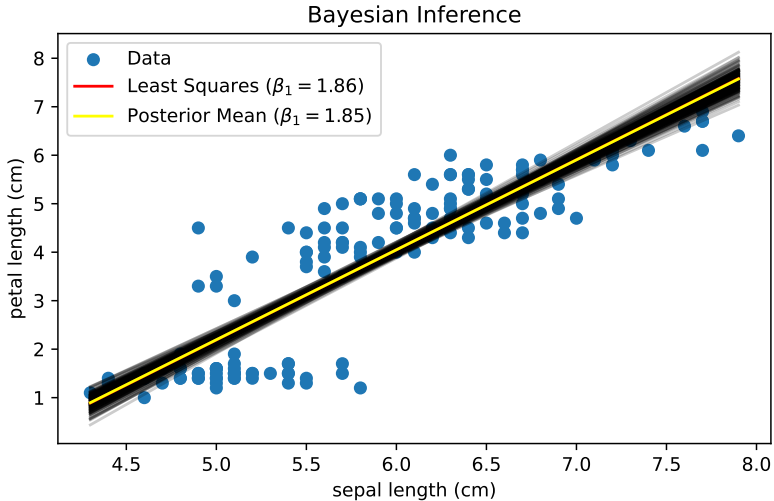
$$\begin{aligned}
\hat{y} &= \beta_0 + \beta_1 x, \\
y &\sim \mathcal{N}(\hat{y}, \sigma), \\
\beta_0 &\sim \mathcal{N}(0, 10^2), \\
\beta_1 &\sim \mathcal{N}(0, 10^2), \\
\sigma &\sim \mathcal{U}(0, 1000).
\end{aligned}$$

Using PyMC3 to "derive" the posterior distribution, we have the following results.

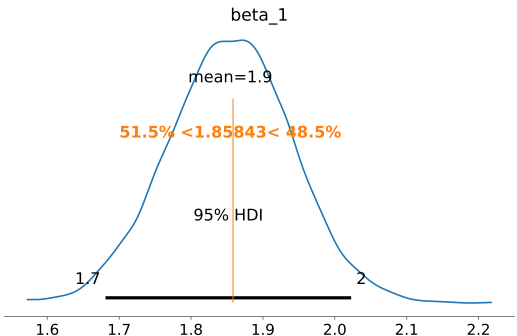
# Bayesian inference - Example 2 (cont.)

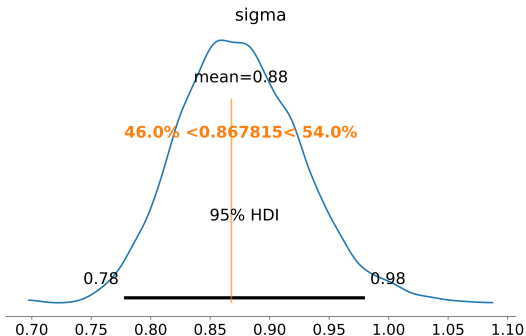
```
with pm.Model() as model:
    beta_0 = pm.Normal("beta_0", mu=0, sigma=10)
    beta_1 = pm.Normal("beta_1", mu=0, sigma=10)
    sigma = pm.Uniform("sigma", lower=0, upper=1000)
    y_hat = beta_0 + beta_1*x
    y_var = pm.Normal("y", mu=y_hat, sigma=sigma,
                      observed=y)
    trace = pm.sample(5000, return_inferencedata=False)
```

# Bayesian inference - Example 2 (cont.)

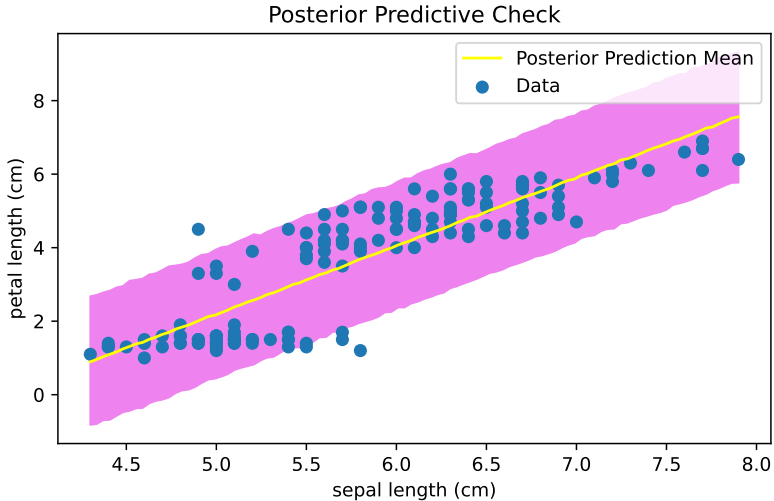


# Bayesian inference - Example 2 (cont.)



# Bayesian inference - Example 2 (cont.)



# Agenda

1. Bayesian inference
2. **Binomial probability**
3. "Deriving" posterior distribution
4. Markov chain Monte Carlo
5. Probabilistic programming
6. Generalized linear model

# Binomial probability

A random variable $Y$ is called the outcome of a **Bernoulli trial** or **Bernoulli distributed** with parameter $p$ $(0 \leq p \leq 1)$, denoted by $Y \sim \text{Bernoulli}(p)$, if

$$\begin{cases} P(Y = 1) = p, \\ P(Y = 0) = 1 - p. \end{cases}$$

$(Y = 1)$ is typically called **"success"**, $(Y = 0)$ is **"failure"**, and $p$ is sucess probability.

A sequence of random variables $Y_1, Y_2, \ldots$ is called a sequence of Bernoulli trials or a **Bernoulli process** if $Y_1, Y_2, \ldots$ are independent and identically Bernoulli$(p)$ distributed.

# Binomial probability

**Problem.** Let $D = \{Y_1 = y_1, Y_2 = y_2, ..., Y_n = y_n\}$ be a sequence of Bernoulli trials, "find" $p$.

**Inference.** Consider a parameter $\theta = p$ which is a continuous random variable and ranges $[0, 1]$.
Given $Y \sim \text{Bernoulli}(\theta)$ with an observation $y \in \{0, 1\}$, the likelihood function for $\theta$ with the data $y$ is

$$p(y|\theta) = \theta^y(1-\theta)^{1-y} = \begin{cases} \theta & y = 1, \\ 1-\theta & y = 0. \end{cases}$$

Then, if the data $D = \left\{y_i\right\}_{i=1}^n$, the likelihood function for $\theta$ with $D$ is

$$p(D|\theta) = \prod_{i=1}^{n} \theta^{y_i}(1-\theta)^{1-y_i} = \theta^{\sum_{i=1}^n y_i}(1-\theta)^{\sum_{i=1}^n (1-y_i)} = \theta^z(1-\theta)^{n-z},$$

where $z = \sum_{i=1}^n y_i$ is the number of successful trials of $n$ trials on dataset $D$.

# Binomial probability (cont.)

"Specifying" the prior distribution for $\theta$: **Beta distribution** with parameters $a, b$ $(a > 0, b > 0)$, $\theta \sim \text{Beta}(a, b)$. Then, we have

$$p(\theta; a, b) \propto \theta^{a-1}(1-\theta)^{b-1}, 0 \leq \theta \leq 1.$$

(More about Beta distribution on `https://en.wikipedia.org/wiki/Beta_distribution`.)

Based on Bayes' rule, the posterior distribution for $\theta$ with the observed data $D$ is

$$p(\theta|D) \propto p(D|\theta)p(\theta; a, b) \propto \theta^{z}(1-\theta)^{n-z}\theta^{a-1}(1-\theta)^{b-1} \propto \theta^{z+a-1}(1-\theta)^{n-z+b-1}.$$

To conclude, $(\theta|D) \sim \text{Beta}(a + z, b + n - z)$.
Thus, Beta distribution is known as **conjugate prior** to the Bernoulli distribution.

# Binomial probability (cont.)

Note that **"hyperparameter"** $a, b$ of the prior distribution and $\kappa = a + b$ are typically called **pseudo-count** since

- Before observing the data $D$, $\theta \sim \text{Beta}(a, b)$, our prior belief in the success probability $\theta$ is such that there were $a$ successes and $b$ failures (in a total of $\kappa = a + b$ trials),
- The data $D$ shows $z$ successes and $n - z$ failures (in a total of $n$ trials),
- After observing the data $D$, $(\theta|D) \sim \text{Beta}(a + z, b + n - z)$, our belief is such that there were $a + z$ sucesses and $b + n - z$ failures (in total of $\kappa + n$ trials).

# Binomial probability (cont.)

Then, we can say that the posterior distribution is always a "compromise" between the prior distribution and the likelihood function, such as

- Mean of the prior distribution $\theta \sim \text{Beta}(a, b)$ is given by

$$E(\theta) = \frac{a}{a+b} = \frac{a}{\kappa},$$

- The likelihood function on data $L(\theta|D) = p(D|\theta) = \theta^z(1-\theta)^{n-z}$ is maximized at

$$\hat{\theta}_{\text{MLE}} = \frac{z}{n},$$

- Mean of the posterior distribution $(\theta|D) \sim \text{Beta}(a+z, b+n-z)$ is given by

$$\begin{aligned} E(\theta|D) &= \frac{a+z}{a+z+b+n-z} = \frac{a+z}{\kappa+n} = \frac{\kappa}{\kappa+n}\frac{a}{\kappa} + \frac{n}{\kappa+n}\frac{z}{n} \\ &= \frac{\kappa}{\kappa+n}E(\theta) + \frac{n}{\kappa+n}\hat{\theta}_{\text{MLE}}. \end{aligned}$$

# Binomial probability - Example

- Given data where $z = 1, n = 10$ (one observed success in a total of 10 trials), the likelihood function is given by

$$p(D|\theta) = \theta^z(1-\theta)^{n-z} = \theta^1(1-\theta)^9$$

maximized at

$$\hat{\theta}_{\text{MLE}} = \frac{z}{n} = \frac{1}{10} = 0.1$$

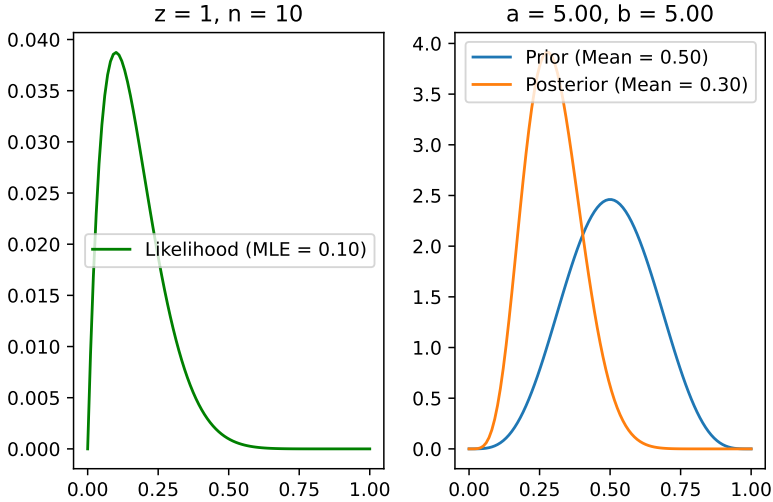
- The prior distribution: $a = 5, b = 5, \kappa = a + b = 10$ (such that there were 5 sucesses in total of 10 trials), $\theta \sim \text{Beta}(a, b) = \text{Beta}(5, 5)$ has

$$E(\theta) = \frac{a}{\kappa} = \frac{5}{10} = 0.5.$$

- The posterior distribution: $a + z = 6, \kappa + n = 20$ (such that there were 6 sucesses in total of 20 trials), $(\theta|D) \sim \text{Beta}(a + z, b + n - z) = \text{Beta}(6, 14)$ has

$$E(\theta|D) = \frac{a+z}{\kappa+n} = \frac{6}{20} = 0.3 = \frac{\kappa}{\kappa+n}E(\theta) + \frac{n}{\kappa+n}\hat{\theta}_{\text{MLE}} = 0.5E(\theta) + 0.5\hat{\theta}_{\text{MLE}}.$$

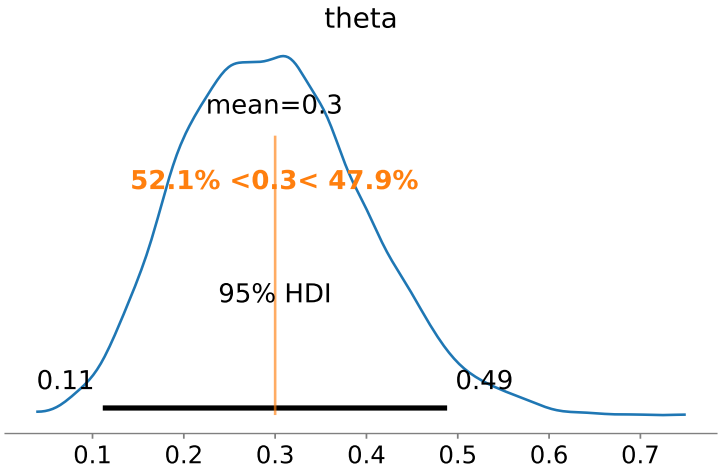
# Binomial probability - Example (cont.)



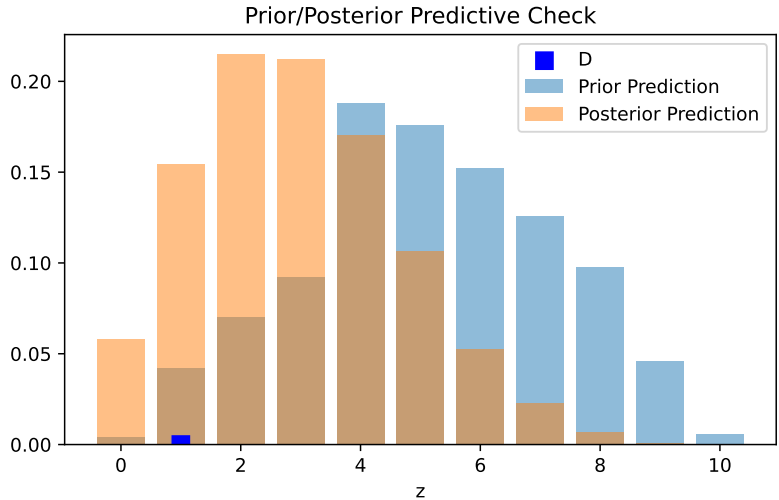
# Binomial probability - Example - PyMC3

```
with pm.Model() as model:
    theta = pm.Beta("theta", alpha=a, beta=b)
    z_var = pm.Binomial("z", p=theta, n=n, observed=z)
    trace = pm.sample(5000, return_inferencedata=False)
    prior_pred = pm.sample_prior_predictive()
    posterior_pred = pm.sample_posterior_predictive(trace)

    az.plot_posterior(trace, var_names=["theta"],
            hdi_prob=0.95, ref_val=posterior_dist.mean())
```

# Binomial probability - Example - PyMC3 (cont.)



# Binomial probability - Example - PyMC3 (cont.)



# Agenda

1. Bayesian inference
2. Binomial probability
3. **“Deriving” posterior distribution**
4. Markov chain Monte Carlo
5. Probabilistic programming
6. Generalized linear model

# “Deriving” posterior distribution

Several methods to “derive” the posterior distribution as follows:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}.$$

- Choosing a $p(\theta)$ **conjugate prior** distribution for the likelihood function $p(D|\theta)$ so that the posterior distribution $p(\theta|D)$ has the same form as the prior distribution.
- “Discreteization” or **“grid approximation”** $\theta$

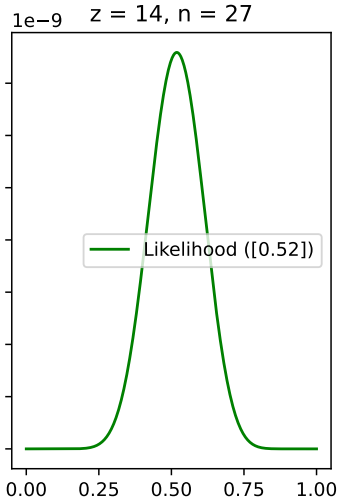
$$p(D) = \sum_{\theta^*} p(D|\theta^*)p(\theta^*).$$

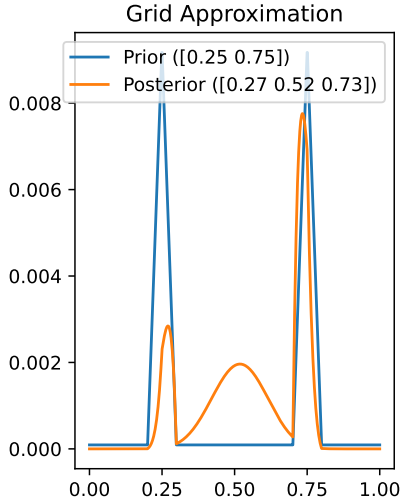
- **Randomly sampling** methods, in which random values are generated, especially **Markov chain Monte Carlo** (MCMC) where the large number of parameter-value $\theta$ from the distribution $p(\theta|D)$ can be generated.
- (other methods)

# Conjugate prior distribution

- `https://en.wikipedia.org/wiki/Conjugate_prior`.
- *Example:* In binomial model, if the likelihood function is Bernoulli (or binomial distribution), we find that the posterior distribution has the same mathematical form as the prior which is both Beta distribution for $\theta = p$ (success probability).

# “Grid approximation” - Example



# Rejection sampling - Example

Consider a simple Bayesian inference problem: we want to infer a $X \sim \text{Exp}(1)$ from an observation $y$ of $Y \sim \mathcal{N}(0, X)$. We need to find the posterior distribution $(X|Y = y)$.

First, the prior distribution of $X$, $X \sim \text{Exp}(1)$, has the density function as follows:

$$p(x) = e^{-x}\mathbb{I}_{[0,\infty)}(x).$$

The likelihood function of $X$ if $Y \sim \mathcal{N}(0, X)$ is known and it takes $y$ as

$$p(y|x) = \frac{1}{\sqrt{2\pi x}} e^{-y^2/(2x)}.$$

Then, based on Bayes' rule, the posterior distribution $(X|Y = y)$ has the following density function

$$p(x|y) \propto p(y|x)p(x)$$

# Rejection sampling - Example (cont.)

Suppose that

$$f(x) = \frac{1}{\sqrt{x}} e^{-y^2/(2x)-x} \mathbb{I}_{[0,\infty)}(x) \propto p(y|x)p(x) \propto p(x|y),$$

we can use the rejection sampling methods to generate samples from the posterior distribution $(X|Y = y)$. Note that the rejection sampling methods can be applied without $f$ standardization.

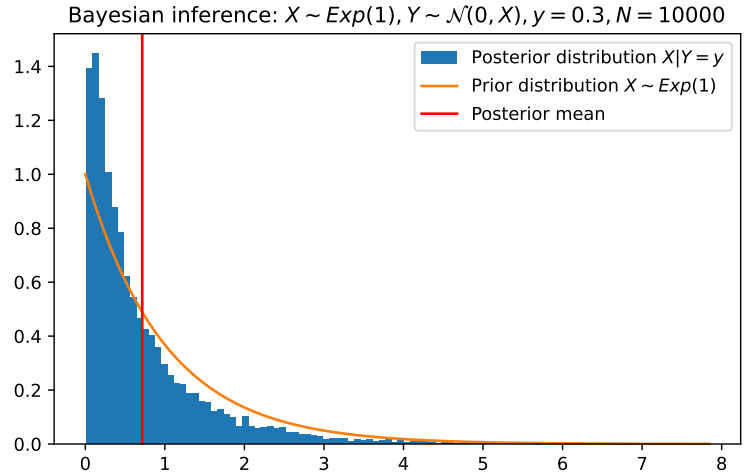
In particular, we can use the envelope rejection sampling algorithm with distribution Exp(1) and a constant

$$c = \frac{1}{|y|} e^{-1/2},$$

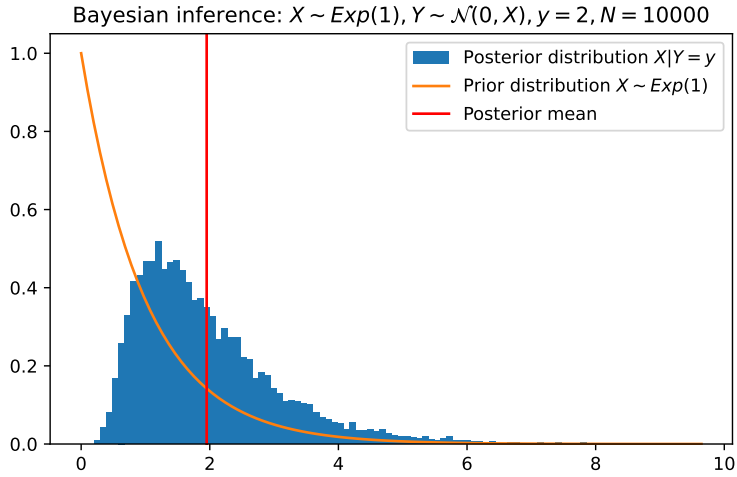
to generate iid $X_1, ..., X_N$ samples from the posterior distribution $(X|Y = y)$, draw histogram and compute

$$E(X|Y = y) \approx \frac{1}{N} \sum_{j=1}^{N} X_j = \bar{X}, \quad Var(X|Y = y) \approx \frac{1}{N-1} \sum_{j=1}^{N} \left(X_j - \bar{X}\right)^2.$$

# Rejection sampling - Example (cont.)



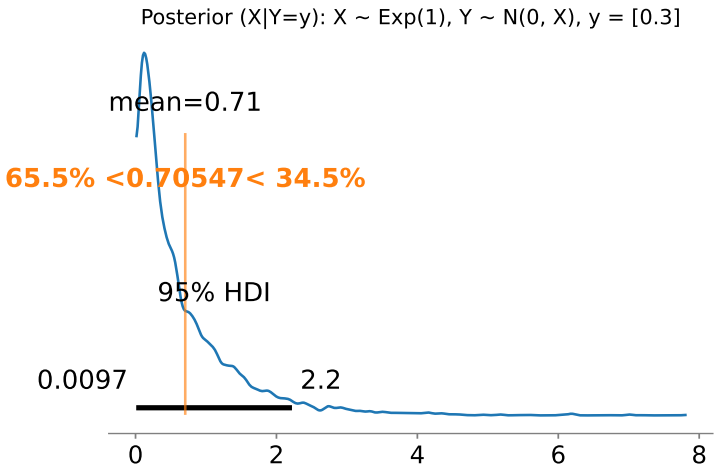
# Rejection sampling - Example (cont.)



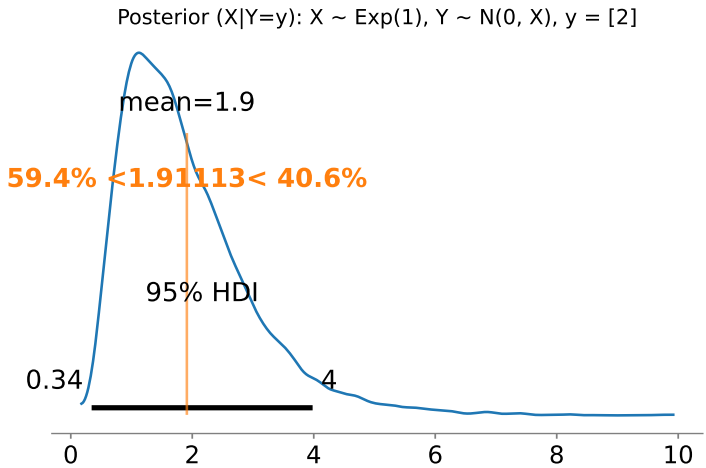
# Rejection sampling - Example - PyMC3

```
#y = [0.3]
#y = [2]
y = [0.3, 2, 1.5, 2.3]

with pm.Model() as model:
    x = pm.Exponential("x", lam=1)
    y_var = pm.Normal("y", mu=0, tau=1/x, observed=y)
    trace = pm.sample(5000, return_inferencedata=False)
    posterior_pred = pm.sample_posterior_predictive(trace)
    az.plot_posterior(trace, var_names=["x"],
            hdi_prob=0.95, ref_val=np.mean(trace["x"]))
```

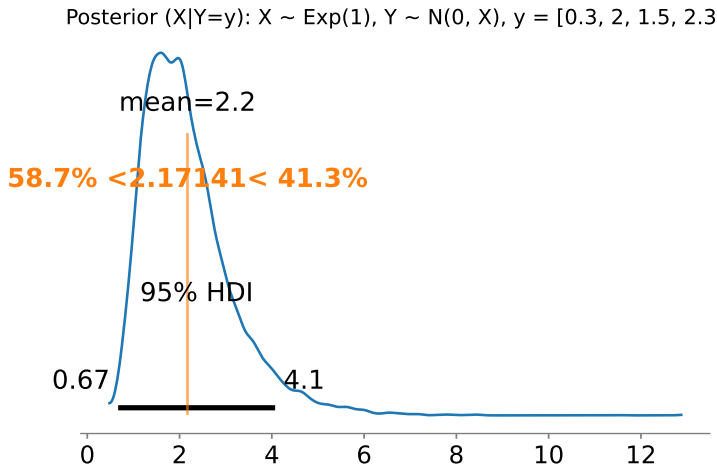
# Rejection sampling - Example - PyMC3 (cont.)



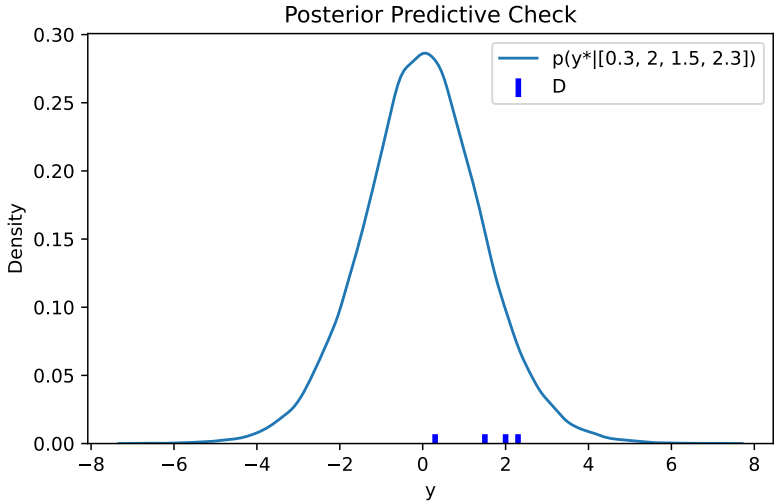
# Rejection sampling - Example - PyMC3 (cont.)



# Rejection sampling - Example - PyMC3 (cont.)



# Rejection sampling - Example - PyMC3 (cont.)



# Agenda

1. Bayesian inference
2. Binomial probability
3. “Deriving” posterior distribution
4. **Markov chain Monte Carlo**
5. Probabilistic programming
6. Generalized linear model

# Markov chain Monte Carlo

- Suppose that we can specify the prior distribution $p(\theta)$ and the likelihood function $p(D|\theta)$ "up to a multiplicative constant", or equivalently, can compute $f(\theta) \propto p(\theta)p(D|\theta) \propto p(\theta|D)$, what the random sampling methods produce for us is representative samples for the posterior distribution $p(\theta|D)$.
- The previous random sampling methods (such as rejection sampling) generate a sequence $\theta_1, \theta_2, ... \overset{\text{iid}}{\sim} \frac{1}{Z_f} f(\theta)$ where $Z_f$ is normalizing constant. In many cases (such as $\theta \in \mathbb{R}^d$ where $d$ is large), these methods are often ineffective.
- **Markov Chain Monte Carlo** (MCMC) generates a **Markov chain** $\theta_1, \theta_2, ...$ whose **stationary distribution** is $\frac{1}{Z_f} f(\theta)$, which makes this method much easier and more effective.
- It is MCMC algorithms (and software, along with fast computer hardware), that allow us to do Bayesian data analysis for realistic datasets and applications.

# Metropolis-Hastings method

**MH algorithm.** (Metropolis-Hastings method)

Input:

- a probability function $f$ ranges $[0, \infty)$ (non-normalised target density),
- a transition density function $g(x'|x)$ (proposal density),
- $X_0$ where $f(X_0) > 0$.

Output: a sample of a Markov chain $X_1, X_2, ...$ with stationary density $\frac{1}{Z_f} f(x)$. Define

$$\alpha(x'|x) = \min\left(1, \frac{f(x')g(x|x')}{f(x)g(x'|x)}\right).$$

1: **for** $n = 1, 2, 3, ...$ **do**
2: generate $X' \sim g(.|X_{n-1})$ #the proposals
3: **if** $\mathcal{U}(0,1) \leq \alpha(X'|X_{n-1})$ **then**
4: $X_n \leftarrow X'$ #$X'$ is accepted with probability $\alpha(X'|X_{n-1})$
5: **else**
6: $X_n \leftarrow X_{n-1}$
7: **end**

# Metropolis-Hastings method - Example

Suppose that we need to sample for an unnormalized density function $f(x) = 2^{-|x|}\mathbb{I}_{\mathbb{Z}}(x)$. Note that

$$Z_f = \sum_{x\in\mathbb{Z}} f(x) = \left(... + \frac{1}{4} + \frac{1}{2} + 1 + \frac{1}{2} + \frac{1}{4} + ...\right) = \left(1 + 2\sum_{x=1}^{\infty} 2^{-x}\right) = 3$$

but $Z_f$ is not required to be computed.

Using MH algorithm, we can easily find a Markov chain that has a stationary distribution as $\frac{1}{Z_f}f(x)$. For this example, we consider a proposal density as:

$$g(X' = x + 1|X = x) = g(X' = x - 1|X = x) = \frac{1}{2}, x \in \mathbb{Z},$$

The acceptance probability is given by

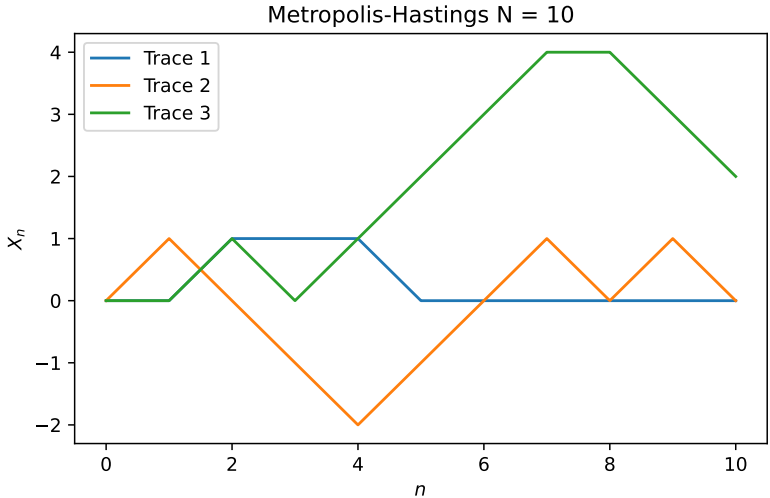
$$\alpha(x'|x) = \min\left(1, \frac{f(x')g(x|x')}{f(x)g(x'|x)}\right) = \min\left(1, \frac{2^{-|x'|}g(x|x')}{2^{-|x|}g(x'|x)}\right) = \begin{cases} 2^{|x|-|x'|} & |x'| > |x| \\ 1 & \text{otherwise} \end{cases}.$$

# Metropolis-Hastings method - Example (cont.)

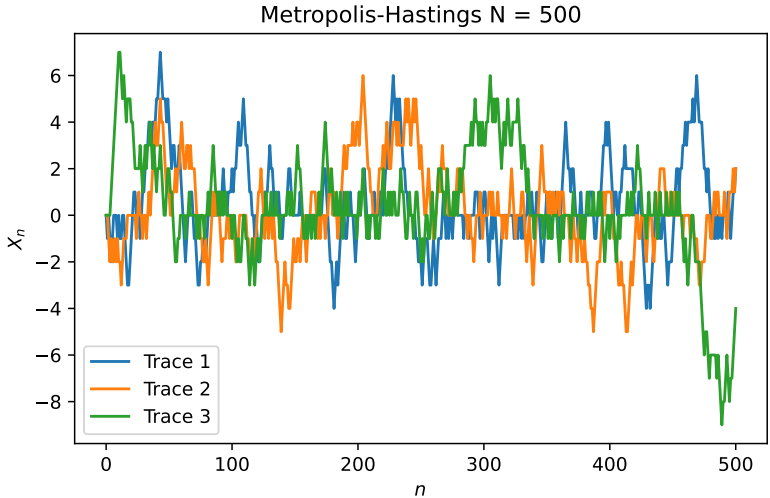
We choose $X_0 = 0$, then substitute the corresponding function $\alpha$ into MH algorithm to get a Markov chain with an unnormalized density function $f(x) = 2^{-|x|}\mathbb{I}_{\mathbb{Z}}(x)$ as its stationary distribution.

1: **for** $n = 1, 2, 3, \ldots$ **do**
2: generate $\epsilon \sim \mathcal{U}\{-1, 1\}$
3: $X' \leftarrow X_{n-1} + \epsilon$
4: generate $U \sim \mathcal{U}(0, 1)$
5: **if** $U \leq 2^{|X_{n-1}|-|X'|}$ **then**
6: $X_n \leftarrow X'$
7: **else**
8: $X_n \leftarrow X_{n-1}$
9: **end**

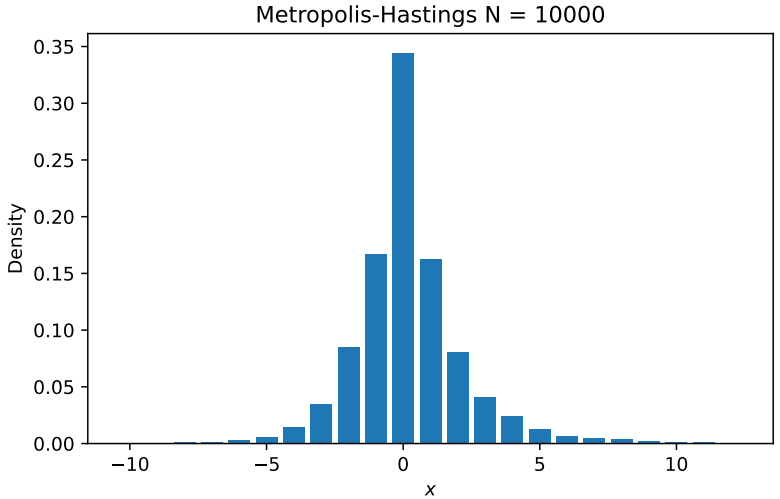
# Metropolis-Hastings method - Example (cont.)



# Metropolis-Hastings method - Example (cont.)



# Metropolis-Hastings method - Example (cont.)



# Metropolis method

If the proposal density is symmetric as:

$$g(x'|x) = g(x|x'), \forall x, x'$$

then we have the acceptance probability is given by

$$\alpha(x'|x) = \min\left(1, \frac{f(x')g(x|x')}{f(x)g(x'|x)}\right) = \min\left(1, \frac{f(x')}{f(x)}\right).$$

The Metropolis-Hastings method for this case is called the **Metropolis method**.
Especially, if the proposals are constructed as

$$X' = X_{n-1} + \epsilon$$

where $\epsilon$ is symmetric distributed(i.e. $\epsilon$ has the same distribution as $-\epsilon$), then it is called **random walk Metropolis**.

# Random walk Metropolis - Example

For discrete state spaces, (such as the preceding example) "increments" or "moves" $\epsilon$ is typically $\mathcal{U}\{-1, 1\}$, or equivalently as

$$P(\epsilon = -1) = P(\epsilon = 1) = \frac{1}{2}.$$

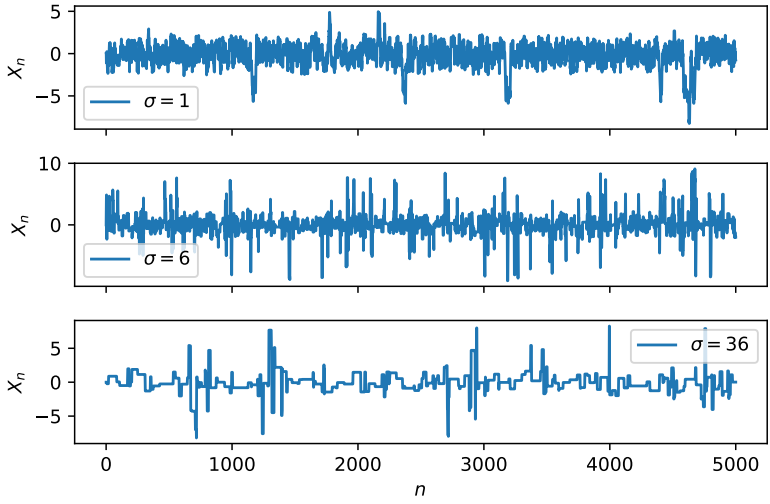
For continuous state space, the most common choice of increments is $\epsilon \sim \mathcal{N}(0, \sigma^2)$ in which the proposal variance $\sigma^2$ has to be chosen to maximise efficiency of the method.

*Example:* Use the random walk Metropolis algorithm to generate samples from the density

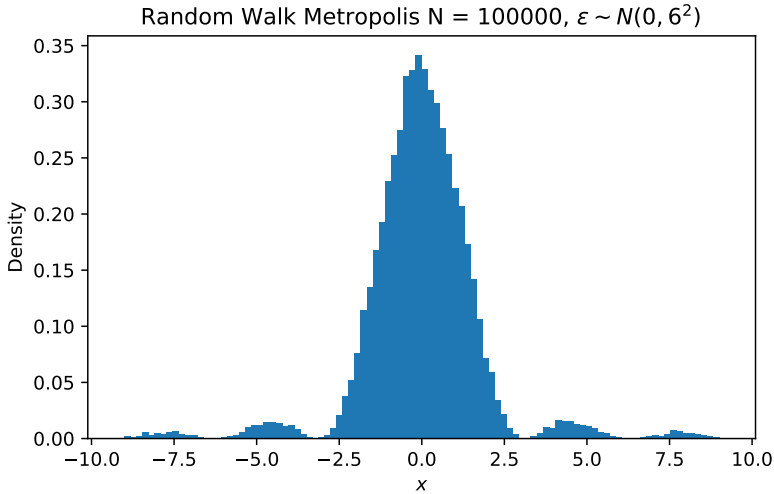
$$f(x) \propto \frac{\sin^2 x}{x^2} \mathbb{I}_{[-3\pi, 3\pi]}(x)$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

# Random walk Metropolis - Example (cont.)



# Random walk Metropolis - Example (cont.)



# Gibbs sampling

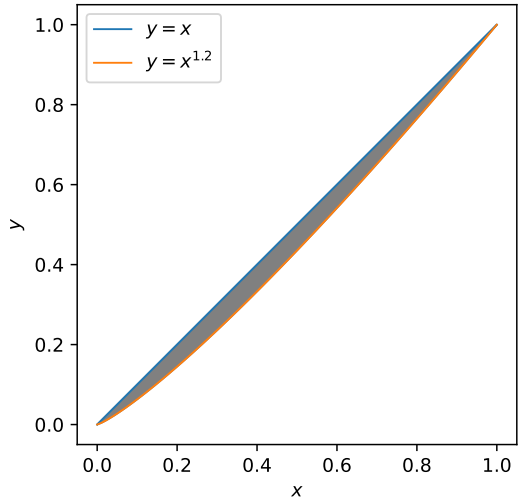
**Gibbs sampling algorithm.** (Gibbs sampling)
Input:

- a distribution $f_{X_1,X_2,...,X_d}$,
- $X^{(0)} = \left(X_1^{(0)}, X_2^{(0)}, ..., X_d^{(0)}\right)$.

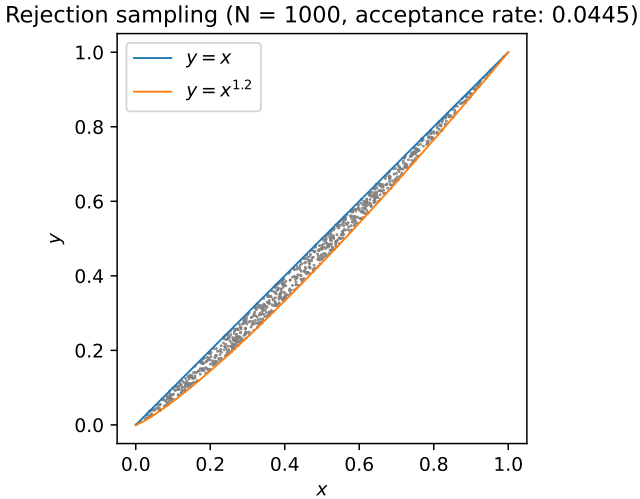
Output: a sample of a Markov chain $X^{(1)}, X^{(2)}, ...$ with stationary distribution $f$.

1: **for** $n = 1, 2, 3, ...$ **do**
2: **for** $i = 1, 2, ..., d$ **do**
3: generate $X_i^{(n)} \sim f_{X_i|\neg X_i}\left(.\middle| X_1^{(n)}, ..., X_{i-1}^{(n)}, X_{i+1}^{(n-1)}, ..., X_d^{(n-1)}\right)$
4: **end for**
5: **end for**

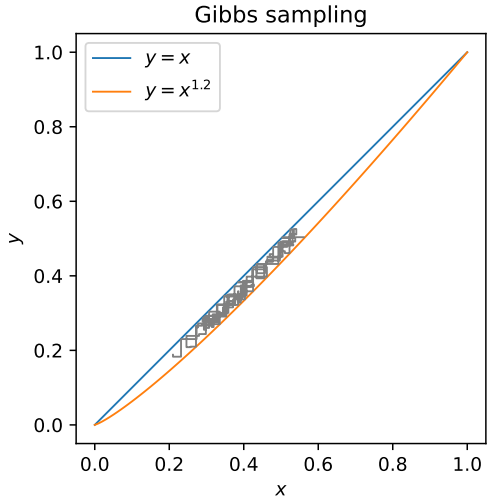
# Gibbs sampling - Example



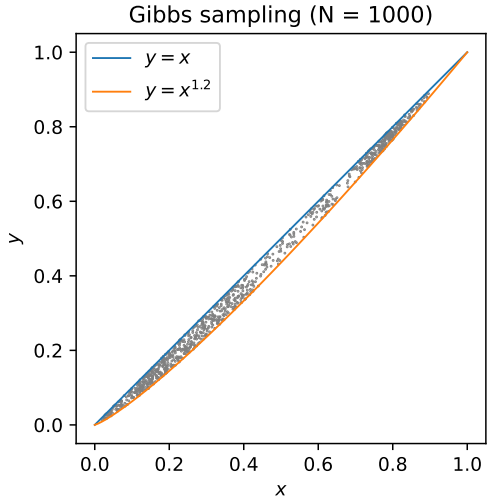
# Gibbs sampling - Example (cont.)



# Gibbs sampling - Example (cont.)



# Gibbs sampling - Example (cont.)



# Agenda

1. Bayesian inference
2. Binomial probability
3. “Deriving” posterior distribution
4. Markov chain Monte Carlo
5. **Probabilistic programming**
6. Generalized linear model

# Probabilistic programming

**Probabilistic programming**

- https://en.wikipedia.org/wiki/Probabilistic_programming.
- Stan, PyMC3/PyMC4, TensorFlow Probability (TFP), Pyro, ...

PyMC3

- https://docs.pymc.io/en/v3/
- Cameron Davidson-Pilon. *Bayesian Methods for Hackers*. Addision-Wesley, 2016. (https://github.com/CamDavidsonPilon/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers)
- Osvaldo Martin. *Bayesian Analysis with Python*. Packt Publishing, 2018.
- Osvaldo A. Martin, Ravin Kumar and Junpeng Lao. *Bayesian Modeling and Computation in Python*. CRC Press, 2022.

# Agenda

1. Bayesian inference
2. Binomial probability
3. “Deriving” posterior distribution
4. Markov chain Monte Carlo
5. Probabilistic programming
6. **Generalized linear model**

# Generalized linear model

- Suppose that we want to determine the dependence of a **predicted variable** (a.k.a dependent/response/output variable) on other **predictor variables**(a.k.a independent/explanatory/input variables).
- Items can be measured on different **scales**, such as metric, count, ordinal, nominal/categorical.
- **Simple linear regression** model

$$y \sim \mathcal{N}(\beta_0 + \beta_1 x, \sigma^2),$$

where $y$ is metric response and $x$ is metric explanatory.

- **Generalized linear model** (GLM)

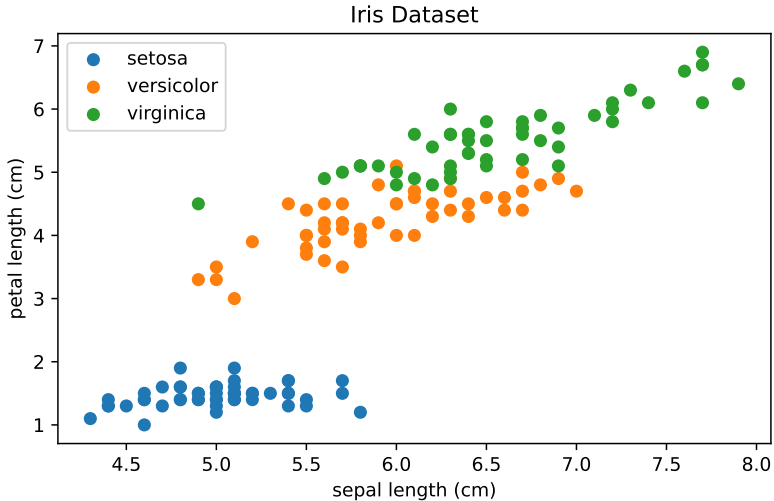
$$\begin{aligned} \text{lin}(x) &= \beta_0 + \beta_1 x_1 + ... + \beta_k x_k, \\ \mu &= f\left(\text{lin}(x), [\text{các tham s}]\right), \\ y &\sim \text{pdf}(\mu, [\text{các tham s}]). \end{aligned}$$

# Generalized linear model (cont.)

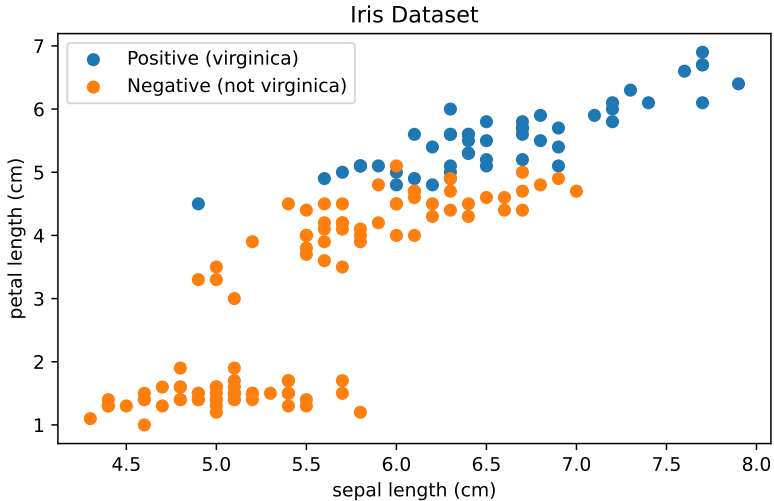
| Scale type of predicted $y$ | Typical noise distribution $y \sim$ pdf $(\mu,$ [parameters]) | Typical inverse link function $\mu = f$ (lin$(x)$, [parameters]) |
|---|---|---|
| **Metric** | $y \sim \text{normal}(\mu, \sigma)$ | $\mu = \text{lin}(x)$ |
| **Dichotomous** | $y \sim \text{bernoulli}(\mu)$ | $\mu = \text{logistic}\,(\text{lin}(x))$ |
| **Nominal** | $y \sim \text{categorical}(\ldots, \mu_k, \ldots)$ | $\mu_k = \dfrac{\exp\,(\text{lin}_k(x))}{\sum_c \exp\,(\text{lin}_c(x))}$ |
| **Ordinal** | $y \sim \text{categorical}(\ldots, \mu_k, \ldots)$ | $\mu_k = \Phi\,((\theta_k - \text{lin}(x))\,/\sigma) - \Phi\,((\theta_{k-1} - \text{lin}(x))\,/\sigma)$ |
| **Count** | $y \sim \text{poisson}(\mu)$ | $\mu = \exp\,(\text{lin}(x))$ |

(John K. Kruschke. *Doing Bayesian Data Analysis – A Tutorial with R, JAGS, and Stan*.)

# Example - Logistic regression



# Example - Logistic regression (cont.)



# Example - Logistic regression (cont.)

**Problem.** From Iris dataset, determine the dependence of class on petal-length and sepal-length, in particular, distinguish virginica class from the others.

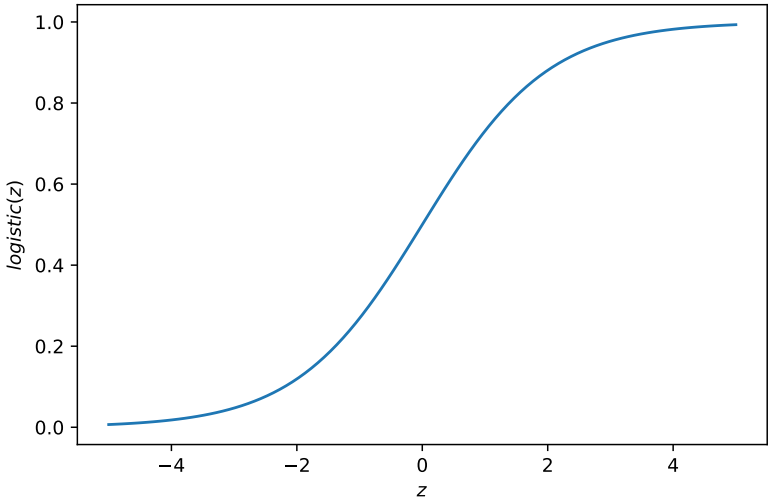
**Inference.** "Based on the scatter plotted data", we form a dependence model of class ($y$, which is equal to 1 if the class is virginica, 0 otherwise) on sepal-length ($x_1$) and petal-length ($x_2$) as follows:

$$\begin{aligned}
\mu &= \text{logistic}(\beta_0 + \beta_1 x_1 + \beta_2 x_2), \\
y &\sim \text{Bernoulli}(\mu), \\
\beta_j &\sim \mathcal{N}(M_j, S_j^2), j = 0, 1, 2.
\end{aligned}$$

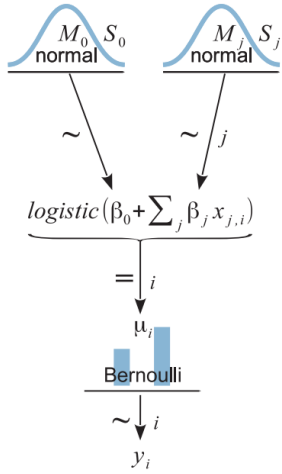
where

$$\text{logistic}(z) = \frac{1}{1 + e^{-z}}.$$

# Example - Logistic regression (cont.)



# Example - Logistic regression (cont.)

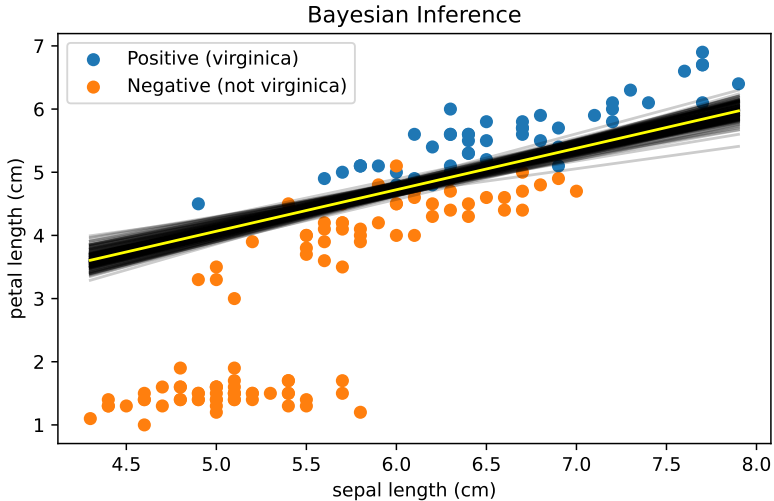



(John K. Kruschke. *Doing Bayesian Data Analysis – A Tutorial with R, JAGS, and Stan*.)

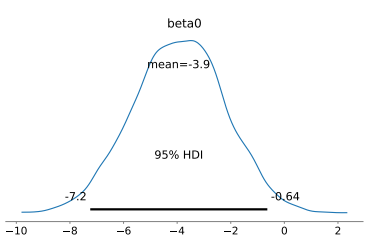
# Example - Logistic regression (cont.)

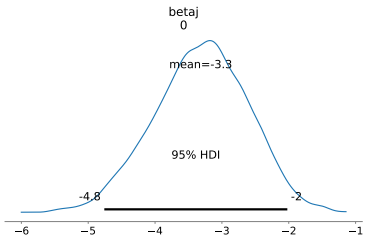
```
with pm.Model() as model:
    beta0 = pm.Normal("beta0", mu=0, sd=2)
    betaj = pm.Normal("betaj", mu=0, sd=2, shape=X.shape
        [1])
    p = pm.invlogit(beta0 + pm.math.dot(X, betaj))
    y_var = pm.Bernoulli("y", p, observed=y)
    trace = pm.sample(5000, return_inferencedata=False)

pm.model_to_graphviz(model)
```

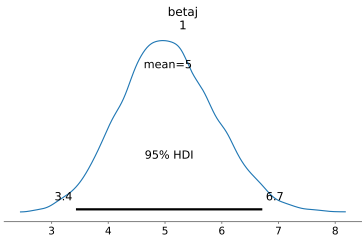
# Example - Logistic regression (cont.)



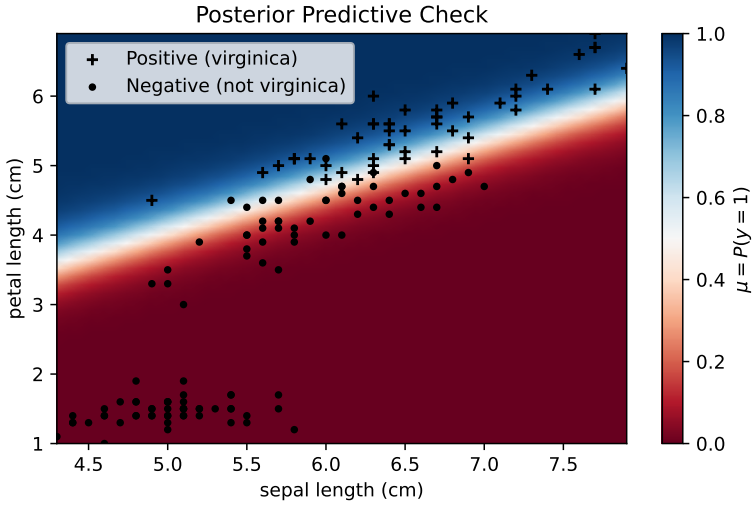
# Example - Logistic regression (cont.)



# Example - Logistic regression (cont.)



# References

**Chapter 2, 6-9, 15, 21.** John K. Kruschke. *Doing Bayesian Data Analysis – A Tutorial with R, JAGS, and Stan*. Elsevier, 2015.

**Chapter 4.** Jochen Voss. *An Introduction to Statistical Computing - A Simulation-based Approach*. John Wiley & Sons, 2014.