# Sampling and MCMC Methods for Bayesian Inference

# Takeaways:

- If you have samples from a distribution, you know a lot about the distribution!
- It's not always straightforward to obtain samples, but there are techniques that can help
- There are accessible computational tools that can help
- We can think geometrically about MCMC sampling
- We can recognize some problems that can emerge from sampling

# Outline

Why sample?

Basic sampling algorithms

      -Inverse cdf

      -Rejection Sampling

MCMC Methods

      -Idea and first example - Metropolis Algorithm

      -Markov chains and stationary distributions

      -Metropolis Hastings

      -Gibbs Sampling

      -Gradient-based sampling (HMC, NUTS)

Introduction to pymc3

Convergence diagnostics

# Samples – what are they good for?

Sometimes it is hard to compute a distribution of interest (like the posterior distribution $p(\theta \mid y)$ in a non-conjugate Bayesian model BUT still possible to generate samples from that distribution
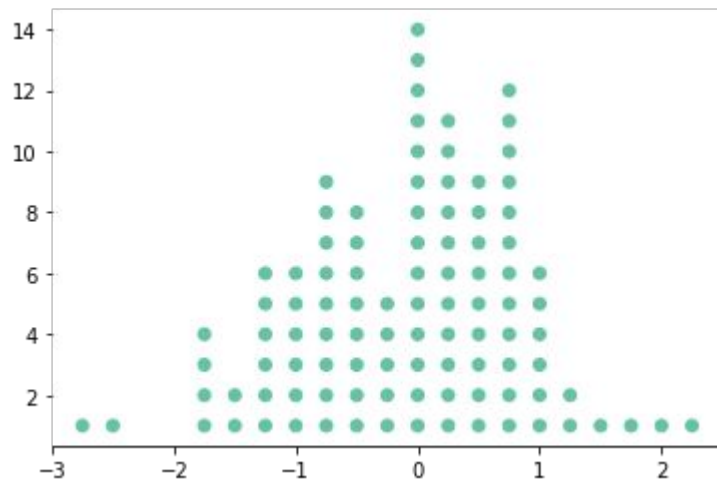
# Samples – what are they good for?

$P(\theta > a)$

$\mathbb{E}[\theta]$

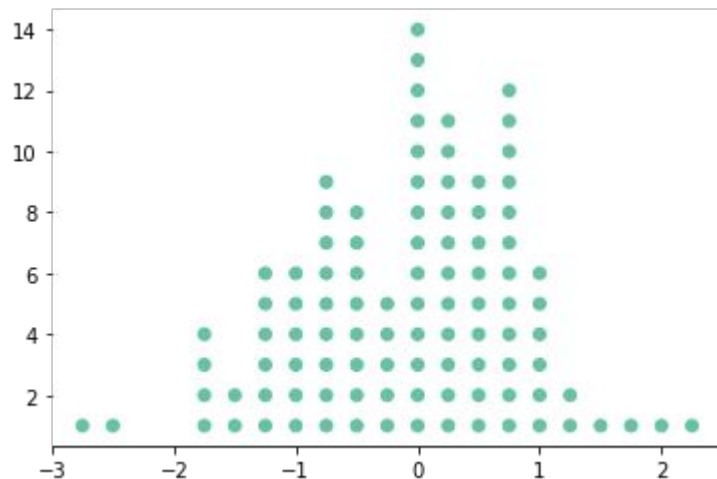$\mathbb{E}[f(\theta)]$

…

# Samples – what are they good for?

$$P(\theta > a) \approx \frac{\#\{\theta_i | \theta_i > a\}}{L}$$

$$\mathbb{E}[\theta] \approx \frac{1}{L} \sum_{i=0}^{L} \theta_i$$

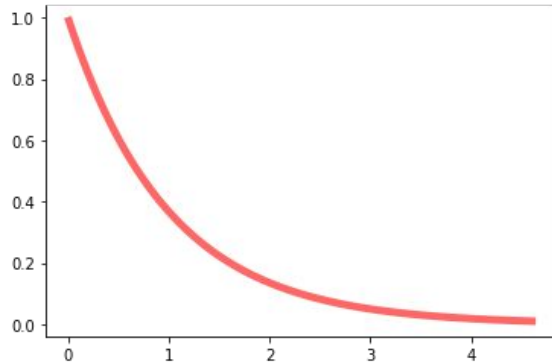$$\mathbb{E}[f(\theta)] \approx \frac{1}{L} \sum_{i=0}^{L} f(\theta_i)$$

# So samples are useful…. How do we get them?

Many methods, from simple ones to various MCMC approaches.

We'll look at a bunch of methods in this workshop! Keep the big goal in mind.

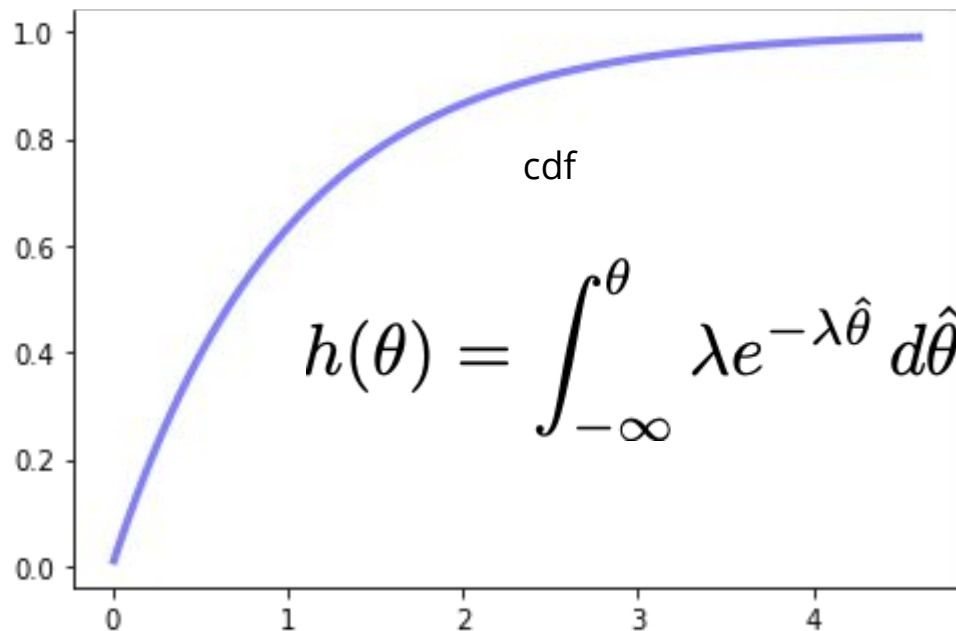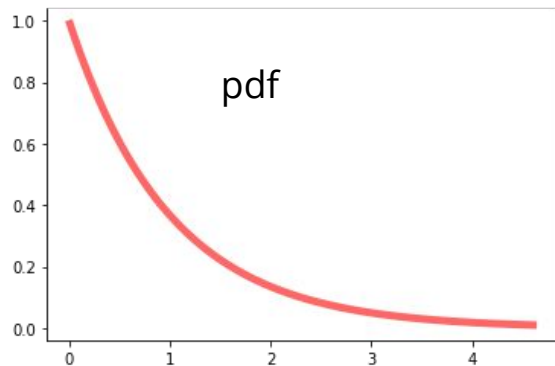# How would you sample from... an exponential distribution?

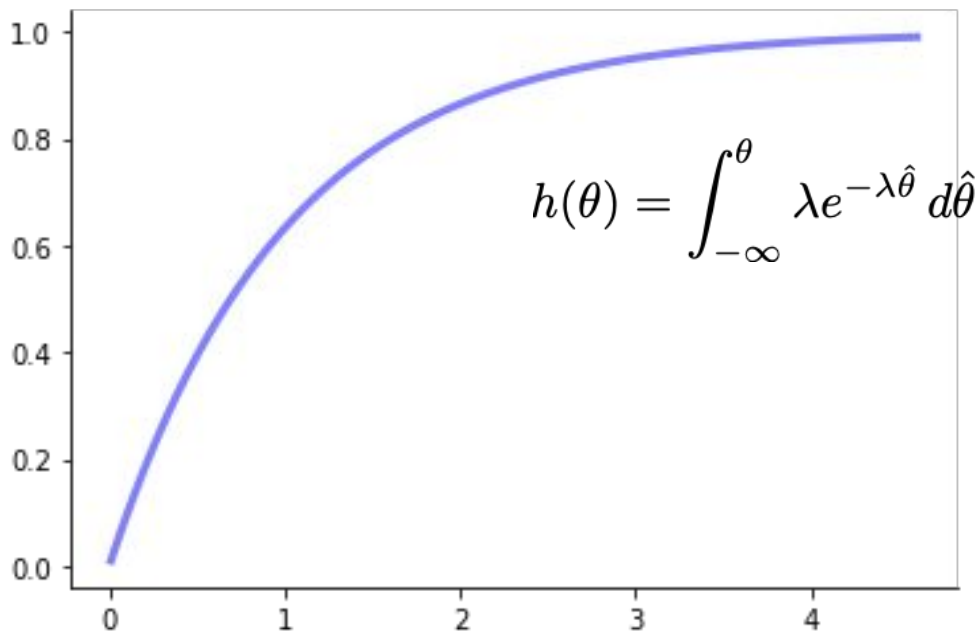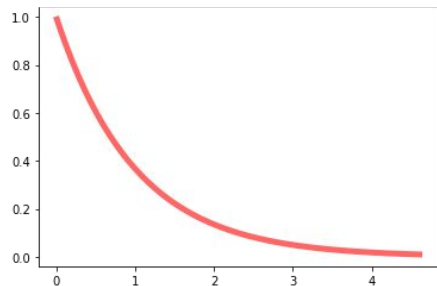$$p(\theta) = \lambda e^{-\lambda\theta}, \theta > 0$$



Assume we can generate (pseudo)-random samples from Uniform(0,1)

# Sampling via the inverse cumulative density function

$$p(\theta) = \lambda e^{-\lambda\theta}, \theta > 0$$

pdf

cdf

$$h(\theta) = \int_{-\infty}^{\theta} \lambda e^{-\lambda\hat{\theta}} d\hat{\theta}$$

Assume we can generate (pseudo)-random samples from Uniform(0,1)

# Sampling via the inverse cumulative density function

$$p(\theta) = \lambda e^{-\lambda \theta}, \theta > 0$$

$$h(\theta) = \int_{-\infty}^{\theta} \lambda e^{-\lambda \hat{\theta}} d\hat{\theta}$$

Assume we can generate (pseudo)-random samples from Uniform(0,1)

# Sampling via the inverse cumulative density function



$$p(\theta) = \lambda e^{-\lambda \theta}, \theta > 0$$

$$u \sim \text{Unif}(0, 1)$$

$$h(\theta) = \int_{-\infty}^{\theta} \lambda e^{-\lambda \hat{\theta}} \, d\hat{\theta}$$

Assume we can generate (pseudo)-random samples from Uniform(0,1)
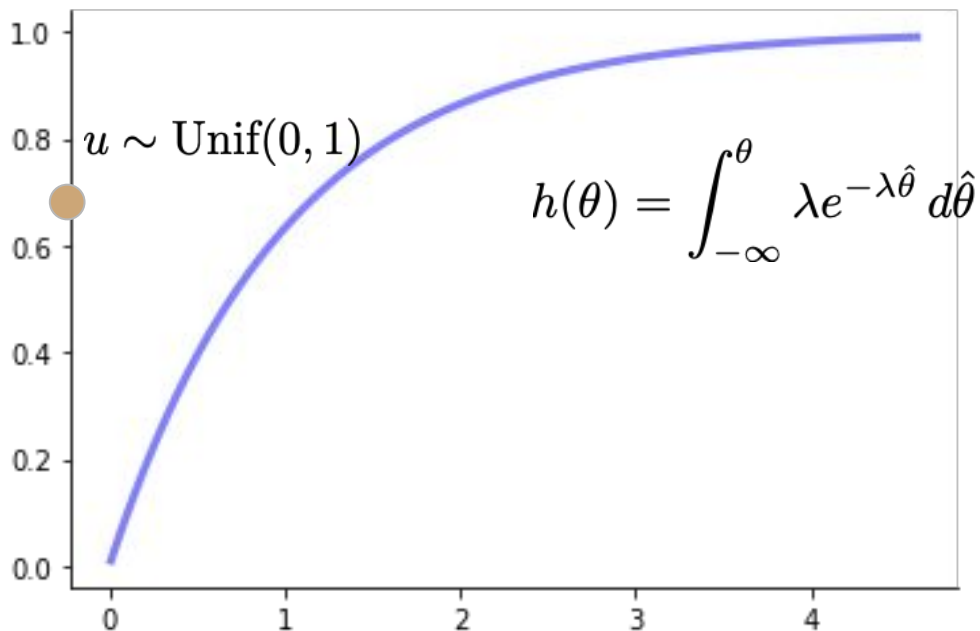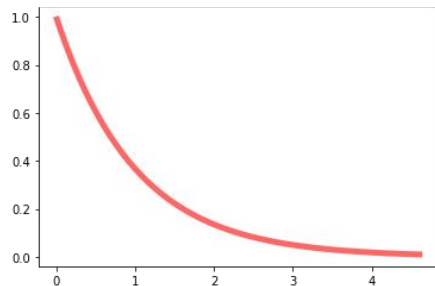
# Sampling via the inverse cumulative density function

$$p(\theta) = \lambda e^{-\lambda \theta}, \theta > 0$$

$$u \sim \text{Unif}(0, 1)$$

$$h(\theta) = \int_{-\infty}^{\theta} \lambda e^{-\lambda \hat{\theta}} \, d\hat{\theta}$$



Assume we can generate (pseudo)-random samples from Uniform(0,1)
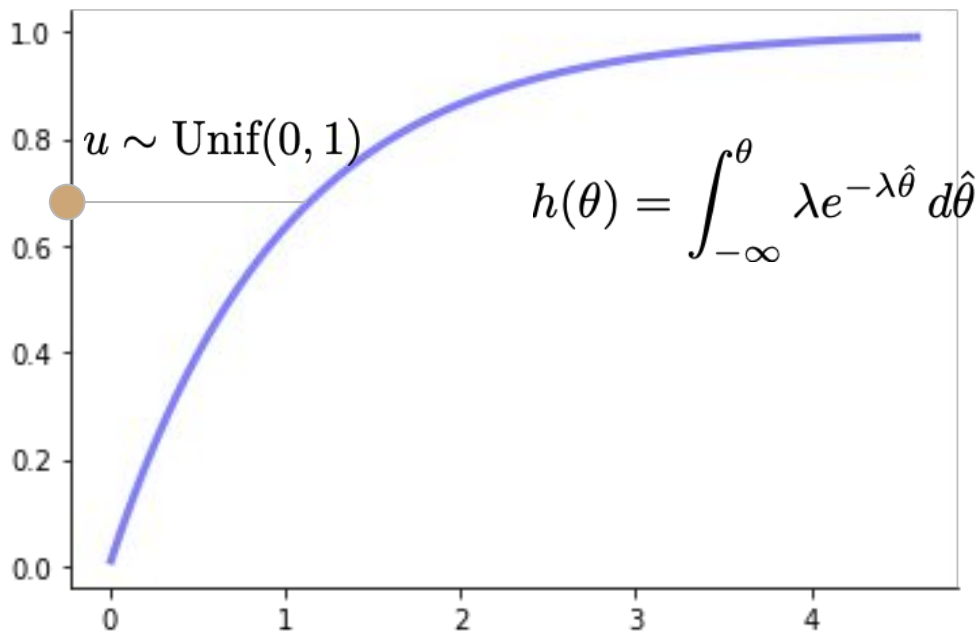
# Sampling via the inverse cumulative density function

$$p(\theta) = \lambda e^{-\lambda\theta}, \theta > 0$$

$u \sim \text{Unif}(0,1)$

$$h(\theta) = \int_{-\infty}^{\theta} \lambda e^{-\lambda\hat{\theta}} \, d\hat{\theta}$$

$h^{-1}(u)$

Assume we can generate (pseudo)-random samples from Uniform(0,1)
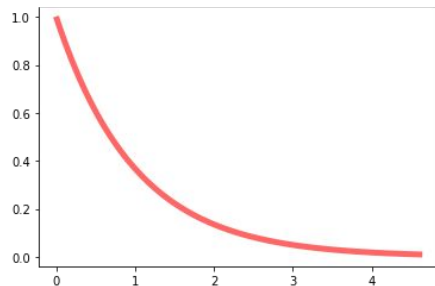
# Sampling via the inverse cumulative density function



$$p(\theta) = \lambda e^{-\lambda\theta}, \theta > 0$$

$$h(\theta) = \int_{-\infty}^{\theta} \lambda e^{-\lambda\hat{\theta}}\, d\hat{\theta}$$

Assume we can generate (pseudo)-random samples from Uniform(0,1)
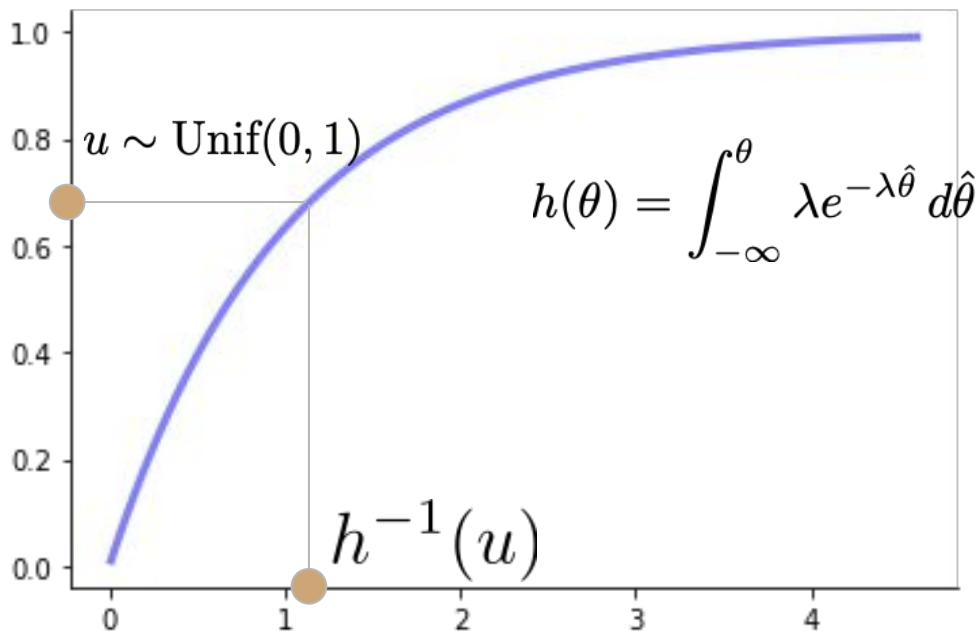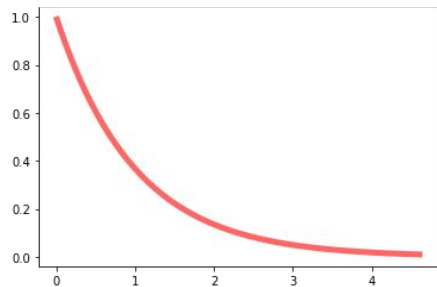
# Sampling via the inverse cumulative density function

$$p(\theta) = \lambda e^{-\lambda \theta}, \theta > 0$$



$$h(\theta) = \int_{-\infty}^{\theta} \lambda e^{-\lambda \hat{\theta}} \, d\hat{\theta}$$

Assume we can generate (pseudo)-random samples from Uniform(0,1)
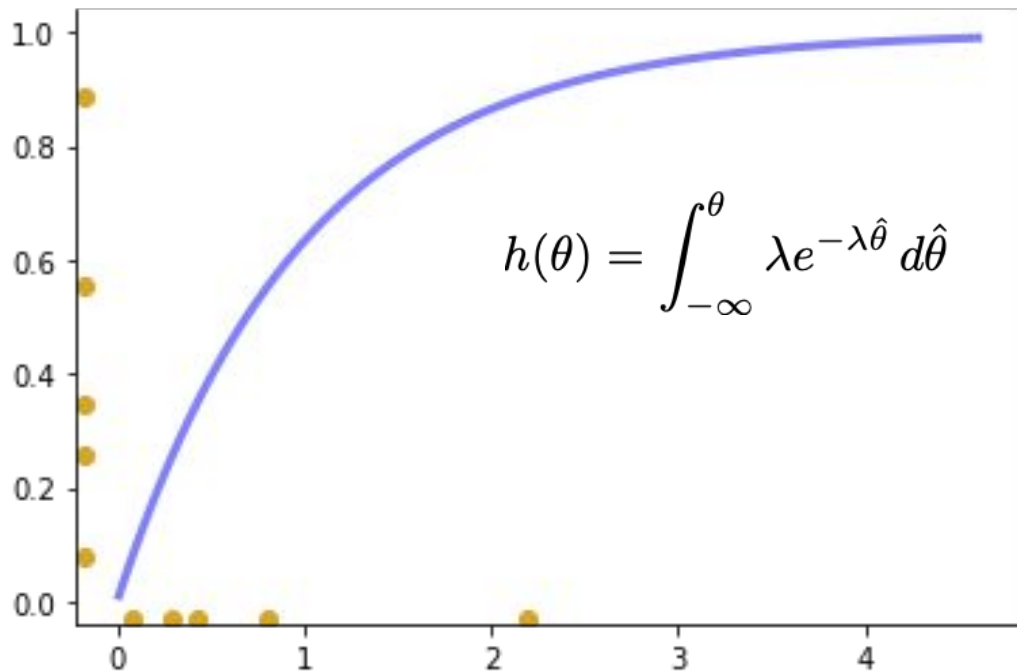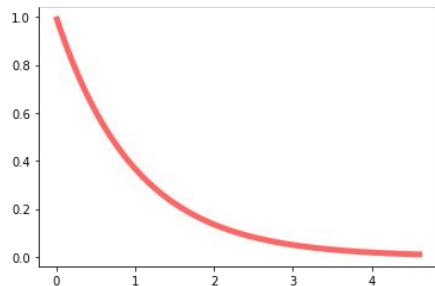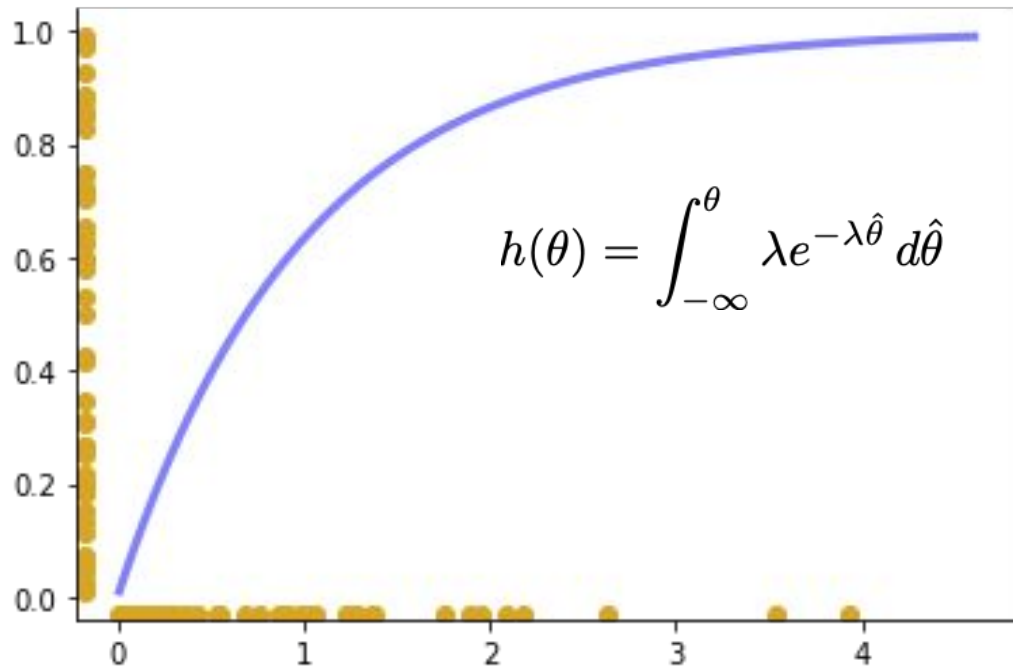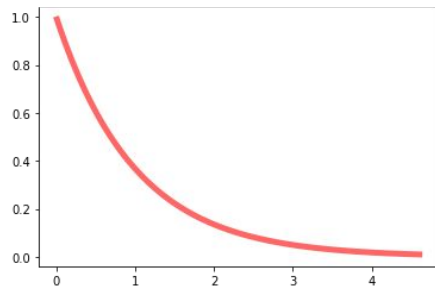
# Sampling via the inverse cumulative density function – summary:

Take random uniform samples $u_i$

Calculate $h^{-1}(u_i)$ where $h(\theta) = \int_{-\infty}^{\theta} p(\hat{\theta}) \, d\hat{\theta}$ . These are the samples!

# For exponential distribution:

$$h(\theta) = \int_0^{\theta} \lambda e^{-\lambda \hat{\theta}} \, d\hat{\theta} = -e^{-\lambda \hat{\theta}}\Big|_0^{\theta} = 1 - e^{-\lambda \theta} \qquad \text{cdf}$$

$$h^{-1}(u) = \lambda^{-1} \log(1 - u) \qquad \qquad \text{Inverse cdf}$$

# Sampling via the inverse cumulative density function – justification

Take random uniform samples $u_i$

Calculate $h^{-1}(u_i)$ where $h(\theta) = \displaystyle\int_{-\infty}^{\theta} p(\hat{\theta})\, d\hat{\theta}$

# Sampling via the inverse cumulative density function – summary:

Take random uniform samples $u_i$

Calculate $h^{-1}(u_i)$ where $h(\theta) = \int_{-\infty}^{\theta} p(\hat{\theta})\, d\hat{\theta}$

# But, we can't always calculate the inverse cdf!

# Rejection Sampling - set up
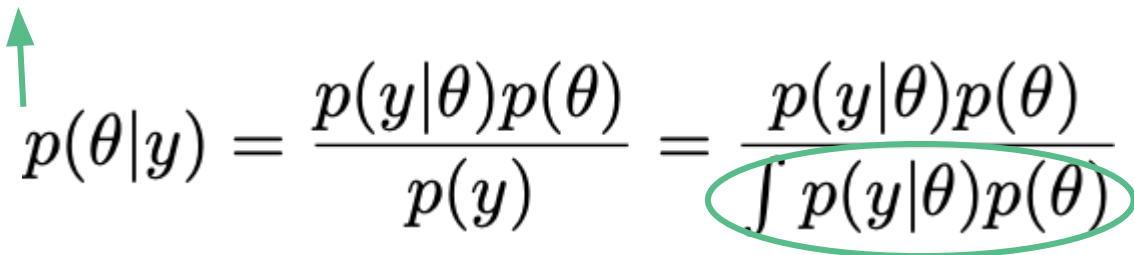
Suppose...
-Want to sample from a distribution with pdf *p(z)*; don't have a simpler approach

-Can evaluate p(z) up to some normalizing constant $Z_p$. That, is can evaluate $\tilde{p}(z)$:

$$p(z) = \frac{1}{Z_p}\tilde{p}(z)$$

Recall: $p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} = \frac{p(y|\theta)p(\theta)}{\int p(y|\theta)p(\theta)}$  probably hard to evaluate

# Rejection Sampling - procedure

$$p(z) = \frac{1}{Z_p}\tilde{p}(z)$$

1. Choose a distribution $q(z)$ that we can sample from and constant k such that $kq(z) > \tilde{p}(z)$ for all $z$
2. Sample $z_0 \sim q(z)$
3. Sample $u_0 \sim \text{Unif}(0, kq(z_0))$
4. Keep $z_0$ as a sample if $u_0 < \tilde{p}(z_0)$, otherwise reject $z_0$

Repeat until have sufficient samples
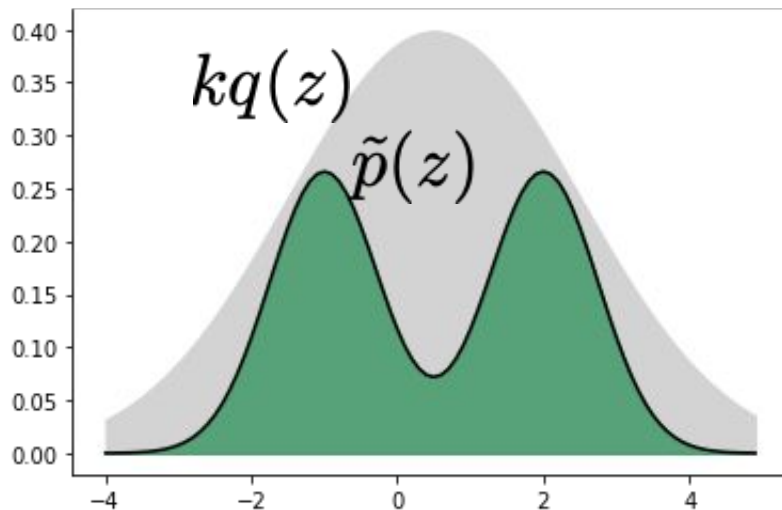
# Rejection Sampling - procedure

$$p(z) = \frac{1}{Z_p}\tilde{p}(z)$$

1. Choose a distribution $q(z)$ that we can sample from and constant k such that $kq(z) > \tilde{p}(z)$ for all $z$
2. Sample $z_0 \sim q(z)$
3. Sample $u_0 \sim$ Unif$( 0, kq(z_0) )$
4. Keep $z_0$ as a sample if $u_0 < \tilde{p}(z_0)$, otherwise reject $z_0$

Repeat until have sufficient samples



21

# Rejection Sampling - procedure

$$p(z) = \frac{1}{Z_p}\tilde{p}(z)$$

1. Choose a distribution $q(z)$ that we can sample from and constant k such that $kq(z) > \tilde{p}(z)$ for all $z$
2. Sample $z_0 \sim q(z)$
3. Sample $u_0 \sim$ Unif$( 0, kq(z_0) )$
4. Keep $z_0$ as a sample if $u_0 < \tilde{p}(z_0)$, otherwise reject $z_0$
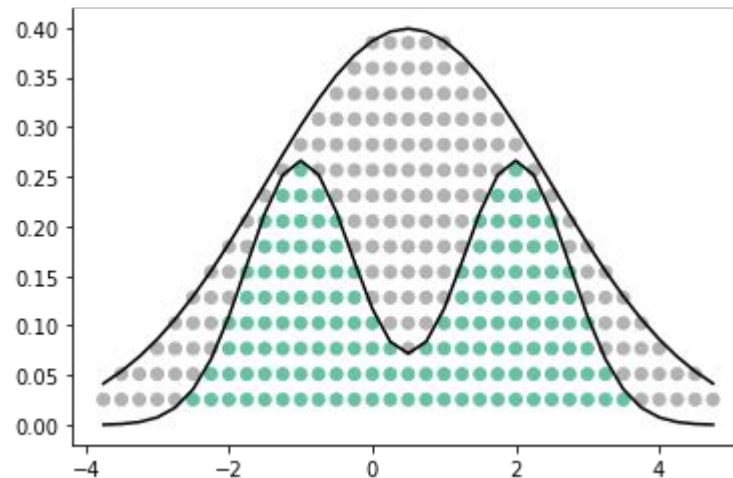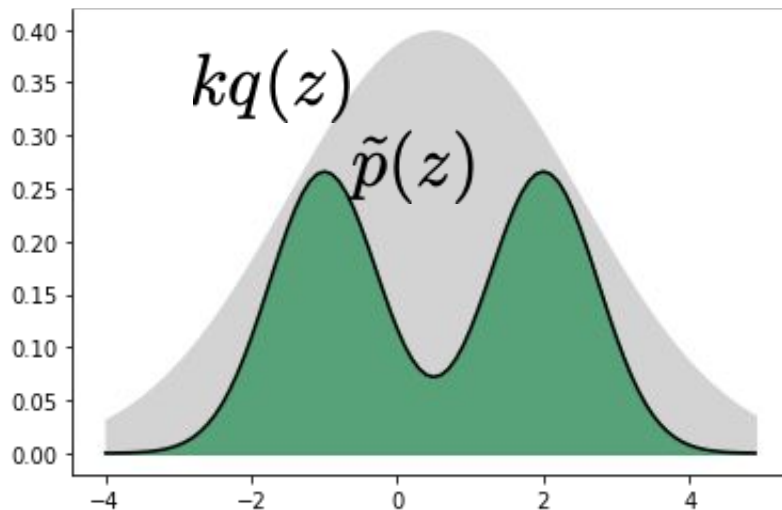
Repeat until have sufficient samples

# Rejection Sampling - procedure

$$p(z) = \frac{1}{Z_p} \tilde{p}(z)$$

1. Choose a distribution $q(z)$ that we can sample from and constant k such that $kq(z) > \tilde{p}(z)$ for all $z$
2. Sample $z_0 \sim q(z)$
3. Sample $u_0 \sim \text{Unif}(0, kq(z_0))$
4. Keep $z_0$ as a sample if $u_0 < \tilde{p}(z_0)$, otherwise reject $z_0$

Repeat until have sufficient samples



23

# Rejection Sampling - procedure

$$p(z) = \frac{1}{Z_p} \tilde{p}(z)$$

1. Choose a distribution $q(z)$ that we can sample from and constant k such that $kq(z) > \tilde{p}(z)$ for all $z$
2. Sample $z_0 \sim q(z)$
3. Sample $u_0 \sim \text{Unif}( 0, kq(z_0) )$
4. Keep $z_0$ as a sample if $u_0 < \tilde{p}(z_0)$, otherwise reject $z_0$
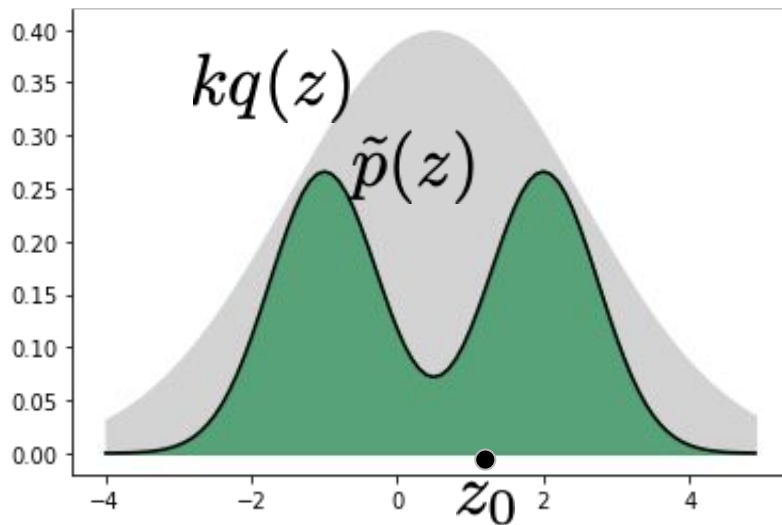
Repeat until have sufficient samples
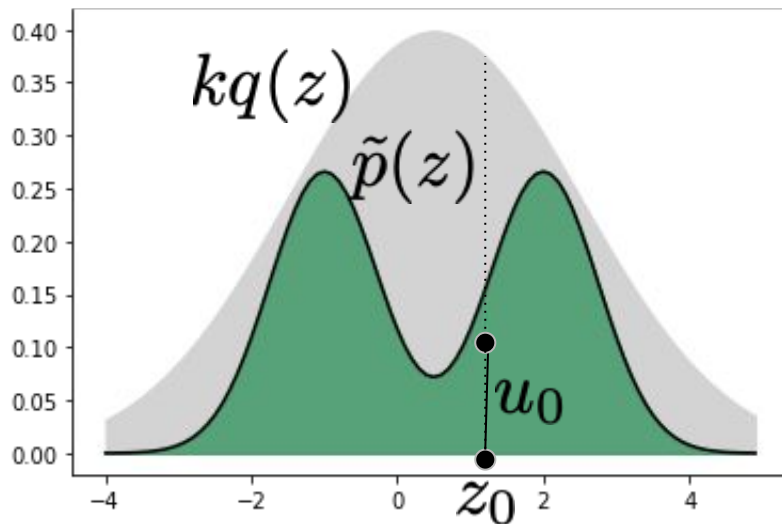
# Rejection Sampling - procedure

$$p(z) = \frac{1}{Z_p}\tilde{p}(z)$$

1. Choose a distribution $q(z)$ that we can sample from and constant k such that $kq(z) > \tilde{p}(z)$ for all $z$
2. Sample $z_0 \sim q(z)$
3. Sample $u_0 \sim \text{Unif}( 0, kq(z_0) )$
4. Keep $z_0$ as a sample if $u_0 < \tilde{p}(z_0)$, otherwise reject $z_0$

Repeat until have sufficient samples
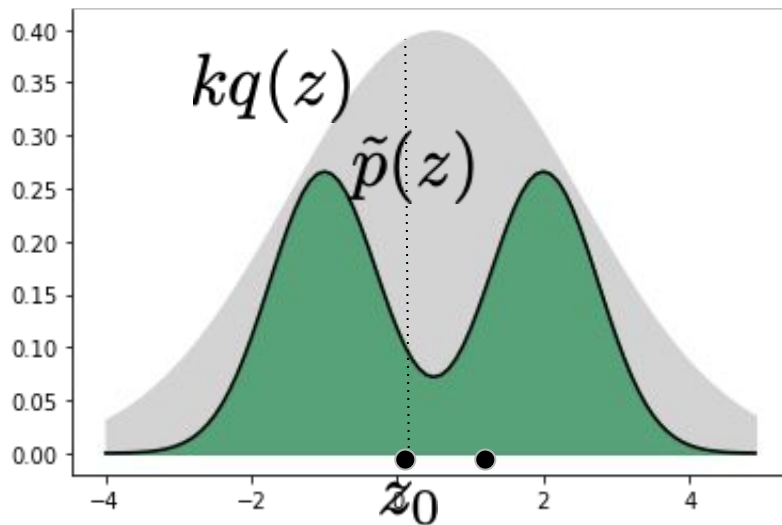
# Rejection Sampling - procedure

$$p(z) = \frac{1}{Z_p}\tilde{p}(z)$$

1. Choose a distribution $q(z)$ that we can sample from and constant k such that $kq(z) > \tilde{p}(z)$ for all $z$
2. Sample $z_0 \sim q(z)$
3. Sample $u_0 \sim$ Unif$( 0, kq(z_0) )$
4. Keep $z_0$ as a sample if $u_0 < \tilde{p}(z_0)$, otherwise reject $z_0$

Repeat until have sufficient samples
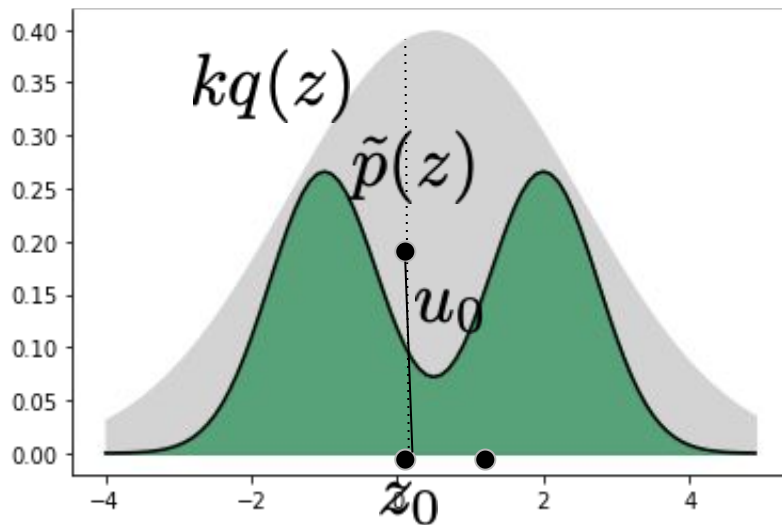
# Rejection Sampling - procedure

$$p(z) = \frac{1}{Z_p} \tilde{p}(z)$$

1. Choose a distribution $q(z)$ that we can sample from and constant k such that $kq(z) > \tilde{p}(z)$ for all $z$
2. Sample $z_0 \sim q(z)$
3. Sample $u_0 \sim \text{Unif}(0, kq(z_0))$
4. Keep $z_0$ as a sample if $u_0 < \tilde{p}(z_0)$, otherwise reject $z_0$

Repeat until have sufficient samples

# Rejection Sampling - Key limitations

- Efficiency depends on good choice of $k$ and $q$
- Becomes inefficient very quickly as the dimensionality of z increases
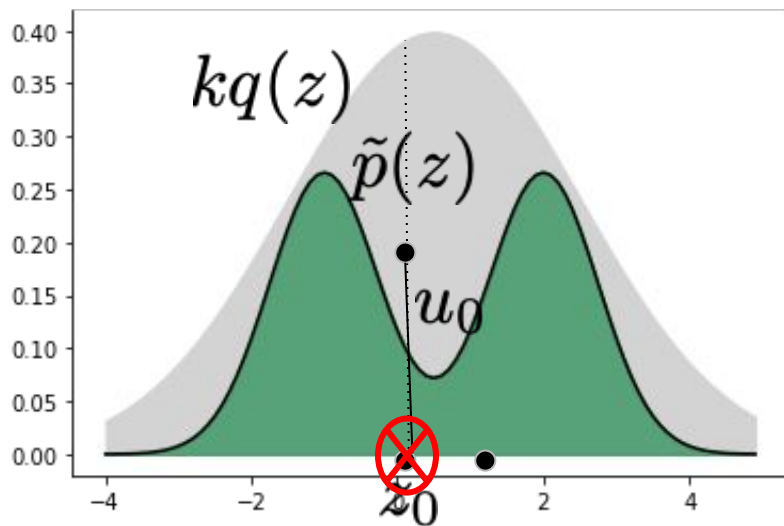
# Markov Chain Monte Carlo

Idea:

-generate one sample based on the previous one (form a Markov Chain)

-do this *cleverly,* so that samples are from distribution of interest
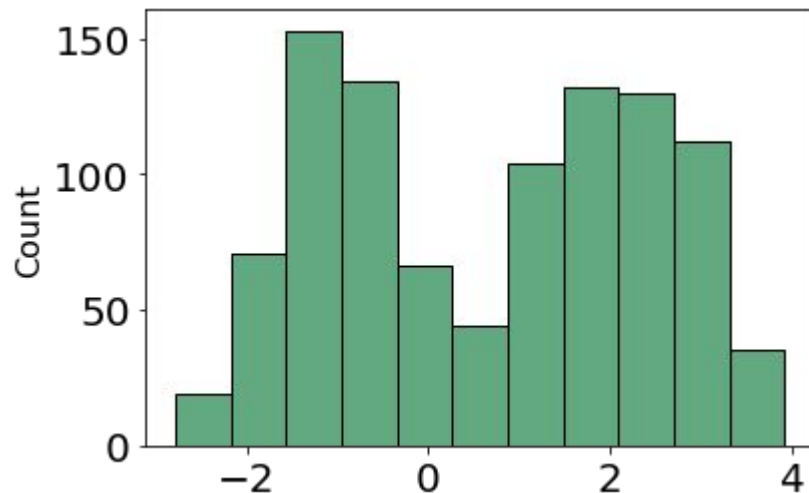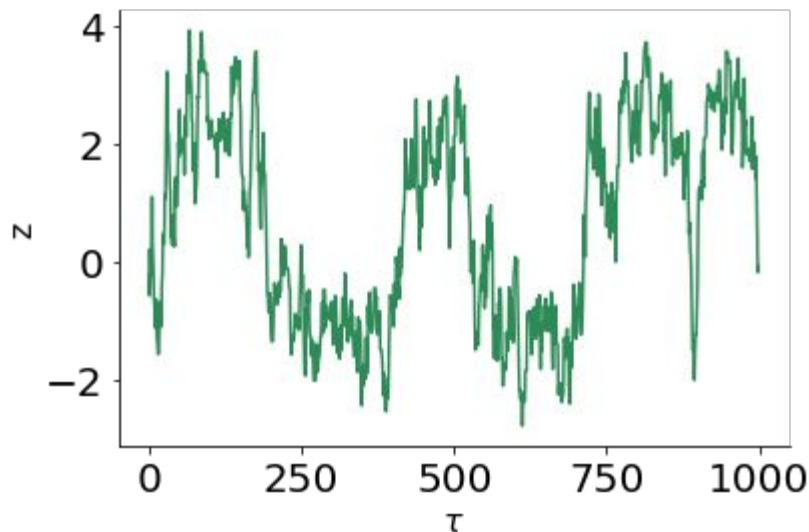
# Markov Chain Monte Carlo – Metropolis Algorithm

Suppose we can evaluate $\tilde{p}(z) = Z_p p(z)$, the unnormalized pdf. Want samples.

Start with $z^0$

Propose a new value for z* according to some ***proposal distribution***

$q(z \mid z^\tau)$ that is symmetric in the sense that $q(z_A \mid z_B) = q(z_B \mid z_A)$.

Accept z* with probability: $A(z^*, z) = \min\left(1, \dfrac{\tilde{p}(z^*)}{\tilde{p}(z^\tau)}\right) = \dfrac{p(z^*)}{p(z^\tau)}$

If accept, set $z^{\tau+1} = z^*$. If not accept, set $z^{\tau+1} = z^\tau$

# Markov Chain Monte Carlo – Metropolis Algorithm

Start with $z^0$

Propose a new value for z* according to some ***proposal distribution***

    $q(z^* \mid z^\tau)$ that is symmetric in the sense that $q(z_A \mid z_B) = q(z_B \mid z_A)$.

Accept z* with probability: $A(z^*, z) = \min\left(1, \dfrac{\tilde{p}(z^*)}{\tilde{p}(z^\tau)}\right) = \dfrac{p(z^*)}{p(z^\tau)}$

If accept, set $z^{\tau+1} = z^*$. If not accept, set $z^{\tau+1} = z^\tau$
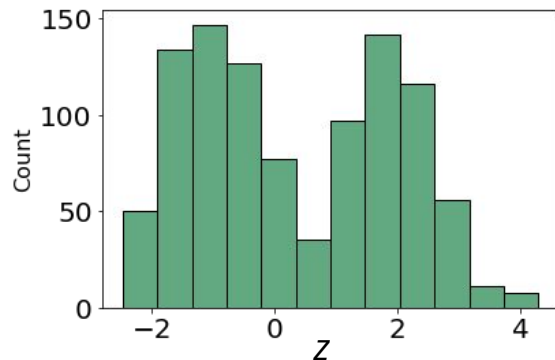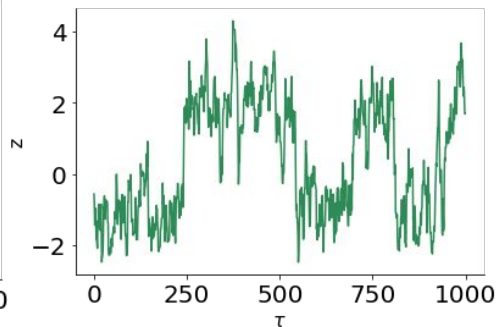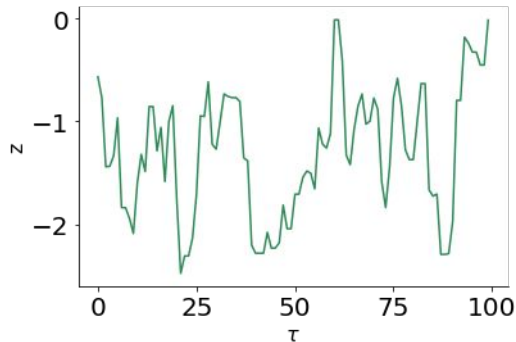
# Markov chains
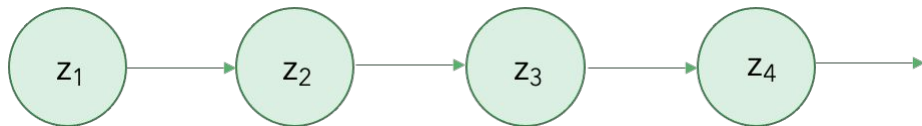


Sequence where future is independent from the past, given the present:

$$p(z^{\tau+1}|z^{\tau}, z^{\tau-1}, ..., z_0) = p(z^{\tau+1}|z^{\tau})$$

We call $T(z^{\tau}, z^{\tau+1}) = p(z^{\tau+1}|z^{\tau})$ the **transition probabilities**

Example: random walk

$$z^{\tau+1} = z^{\tau} + s, \ s \sim N(0, \sigma^2)$$

# Markov chains



A Markov chain has a **stationary distribution** $\pi$ if:

$$\pi(z) = \sum_{z'} T(z', z)\pi(z')$$

In matrix notation:

$$\pi = \pi P \quad \text{where}$$

$$P_{ij} = p(z^{\tau+1} = j | z^{\tau} = i)$$

or, in continuous state-space Markov chain, if:

$$\pi(z) = \int T(z', z)\pi(z')\, dz'$$

# Markov chains

In MCMC methods, we want to sample from some target distribution p

We construct our Markov process so that it will asymptotically reach p as its stationary distribution.

What do we need to show to guarantee that this occurs?

1. That p is stationary distribution. A sufficient condition (that's often easier to work with than the definition of a stationary distribution) is that **detailed balance** holds:

$$\pi(z)T(z, z') = \pi(z')T(z', z)$$

2. The stationary distribution is unique. This holds if the Markov chain satisfies the is (roughly stated): *aperiodic* - gcd of number of steps to return to a state with positive probability is 1 and *positive recurren*t - expected return time to a state is finite. For continuous state space, replace state with set of states with positive probability )

# MCMC – Metropolis Hastings Algorithm

*Generalizes Metropolis algorithm by allowing non-symmetric proposal distributions*

Start with $z^0$

Propose a new value for z* according to some **proposal distribution**

   q(z* | $z^\tau$ ), which is not necessarily symmetric

Accept z* with probability: $A(z^*, z) = \min\left(1, \dfrac{\tilde{p}(z^*)q(z^\tau | z^*)}{\tilde{p}(z^\tau)q(z^* | z^\tau)}\right)$

If accept, set $z^{\tau+1}$ = z*. If not accept, set $z^{\tau+1}$ = $z^\tau$

# MCMC – Metropolis Hastings

recall:
$$A(z^*, z) = \min\left(1, \frac{\tilde{p}(z^*)q(z^\tau|z^*)}{\tilde{p}(z^\tau)q(z^*|z^\tau)}\right)$$

Proving detailed balance: show $p(z)T(z, z') = p(z')T(z', z)$

$$
\begin{aligned}
p(z)T(z, z') &= p(z)q(z'|z)A(z', z) \\
&= \min(p(z)q(z'|z), p(z')q(z|z')) \\
&= \min(p(z')q(z|z'), p(z)q(z'|z)) \\
&= p(z')q(z|z')A(z, z') \\
&= p(z')T(z', z)
\end{aligned}
$$

# MCMC – Metropolis Hastings

-Step size can be a challenge! If we choose it to be too small, we may explore too slowly to be efficient. If too large, we may reject too many samples to be efficient. Want acceptance ratio to be neither too large nor too small.

-Also, target distribution can have different standard deviations in different directions, another challenge if we use something like an isotropic Gaussian as proposal distribution

-We may want to discard the earliest samples in the chain ("burn-in" or "warm-up")

# MCMC – Gibbs Sampling

-Iterate through each parameter, fixing all other parameters and sample from the conditional.

$$z_1^{\tau+1} \sim p(z_1 | z_2^\tau, z_3^\tau ... z_m^\tau)$$

$$z_2^{\tau+1} \sim p(z_2 | z_1^{\tau+1}, z_3^\tau ... z_m^\tau)$$

$$...$$

$$z_m^{\tau+1} \sim p(z_m | z_1^{\tau+1}, z_2^{\tau+1} ... z_{m-1}^{\tau+1})$$

note that we're also typically conditioning on data
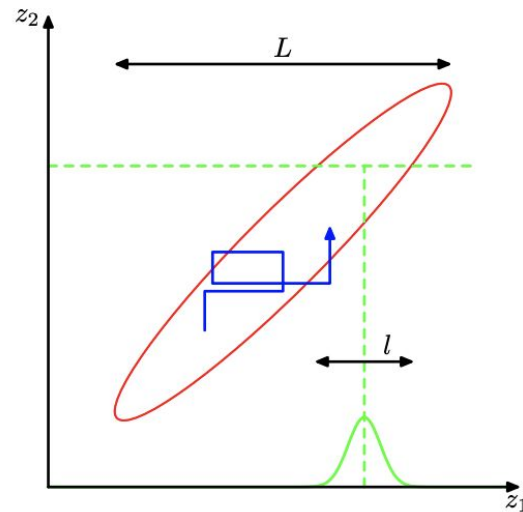


Fig 11.11 from *Pattern Recognition and Machine Learning*, Christopher Bishop

# MCMC – Gibbs Sampling

Conditionals are sometimes nice to calculate in hierarchical models…. e.g.:
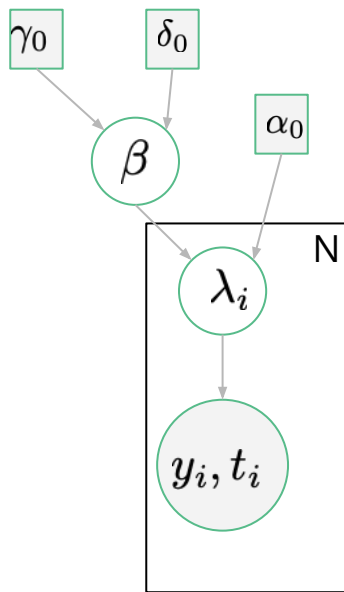


$$\beta \sim \text{Gamma}(\gamma_0, \delta_0)$$

$$\lambda_i \sim \text{Gamma}(\alpha_0, \beta), \ i = 0, ..., N$$

$$y_i \sim \text{Poisson}(\lambda_i t_i) \ i = 0, ..., N$$

hyperparameters: $\gamma_0, \delta_0, \alpha_0$

observations: $y_i, t_i$

want to know about parameters: $\beta, \lambda_i$

$$\beta \sim \text{Gamma}(\gamma_0, \delta_0)$$
$$\lambda_i \sim \text{Gamma}(\alpha_0, \beta),\ i = 0, ..., N$$
$$y_i \sim \text{Poisson}(\lambda_i t_i)\ i = 0, ..., N$$

hyperparameters: $\gamma_0, \delta_0, \alpha_0$

observations: $y_i, t_i$

want to know about parameters: $\beta, \lambda_i$

$$\beta | \lambda, y, t \sim \text{Gamma}(N\alpha_0 + \gamma_0, \sum_i \lambda_i + \delta_0)$$
$$\lambda_i | \beta, \lambda_{/i}, \mathbf{y}, \mathbf{t} \sim \text{Gamma}(y_i + \alpha_0, t_i + \beta)$$

Gamma and poisson pdfs
$$\text{Gamma}(x|a, b) = b^a x^{a-1} e^{-bx} / \Gamma(a)$$
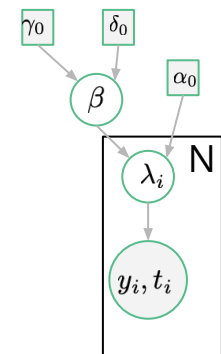$$\text{Poisson}(y|\lambda) = e^{-\lambda} \lambda^y / y!$$

likelihood    prior
$$p(\beta, \lambda | \mathbf{y}, \mathbf{t}) \propto p(\mathbf{y}, \mathbf{t} | \lambda) p(\lambda | \alpha_0, \beta) p(\beta | \gamma_0, \delta_0)$$
$$= \frac{\prod_i (e^{-\lambda_i t_i}(\lambda_i t_i)^{y_i}) \prod_i (\beta^\alpha \lambda_i^{\alpha_0-1} e^{-\beta \lambda_i}) \delta_0^{\gamma_0} \beta^{\delta_0-1} e^{-\delta_0 \beta}}{\Gamma(\alpha_0) \Gamma(\gamma_0) \prod_i y_i}$$
$$\propto e^{-\sum_i (\lambda_i t_i + \beta \lambda_i) + \delta_0 \beta} \beta^{N\alpha_0 + \gamma_0 - 1} \prod \lambda_i^{y_i + \alpha_0 - 1}$$

Combining terms and dropping factors that don't depend on beta or lambda

$$p(\beta | \lambda, y, t) \propto e^{-(\sum_i \lambda_i + \delta_0)\beta} \beta^{N\alpha_0 + \gamma_0 - 1}$$
$$\implies \beta | \lambda, y, t \sim \text{Gamma}(N\alpha_0 + \gamma_0, \sum_i \lambda_i + \delta_0)$$

$$p(\lambda_i | \lambda_{/i}, \beta, y, t) \propto e^{-(t_i + \beta)\lambda_i} \lambda_i^{y_i + \alpha_0 - 1}$$
$$\implies \lambda_i | \lambda_{/i}, \beta, y, t \sim \text{Gamma}(y_i + \alpha_0, t_i + \beta)$$

40

# MCMC – Gibbs Sampling – additional notes

-Can be an efficient choice if if you can actually get all the conditionals you need

-Gibbs sampling is actually a special case of Metropolis Hastings (see e.g PRML , Bishop, Section 11.3)

-Augmenting by including auxiliary variables can be (surprisingly?) useful, to make sampling easier

-Could use Metropolis-within-Gibbs if one or some of the conditionals are hard to sample from

# Gradient-based MCMC methods – Hamiltonian MC

-Want a method that can propose samples that are far away from current, BUT still have a high probability of acceptance

-Will introduce some auxiliary variables and use Hamiltonian dynamics to generate proposals (parameters as **position**, auxiliary **momentum** variables)

-Flexible and efficient sampling method for continuous distributions. Need to be able to evaluate (log) density function (up to normalizing constant), and to compute partial derivatives of the log density

**Random−walk Metropolis**     **Hamiltonian Monte Carlo**
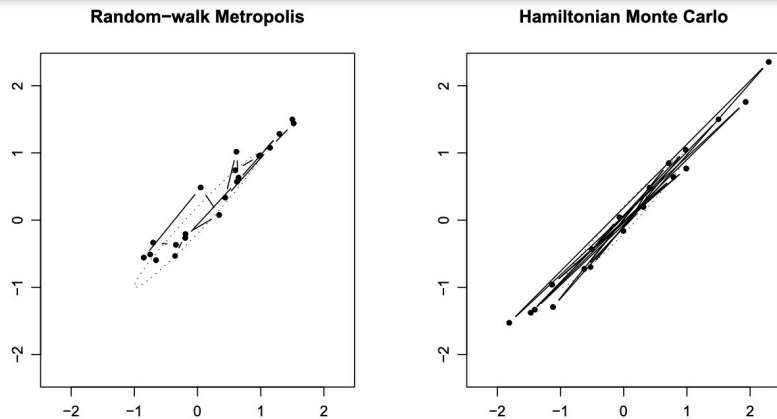
*Figures 4, 5 from Neal, 2011*

Figure 4: Twenty iterations of the random-walk Metropolis method (with 20 updates per iteration) and of the Hamiltonian Monte Carlo method (with 20 leapfrog steps per trajectory) for a 2D Gaussian distribution with marginal standard deviations of one and correlation 0.98. Only the two position coordinates are plotted, with ellipses drawn one standard deviation away from the mean.



**Random−walk Metropolis**     **Hamiltonian Monte Carlo**
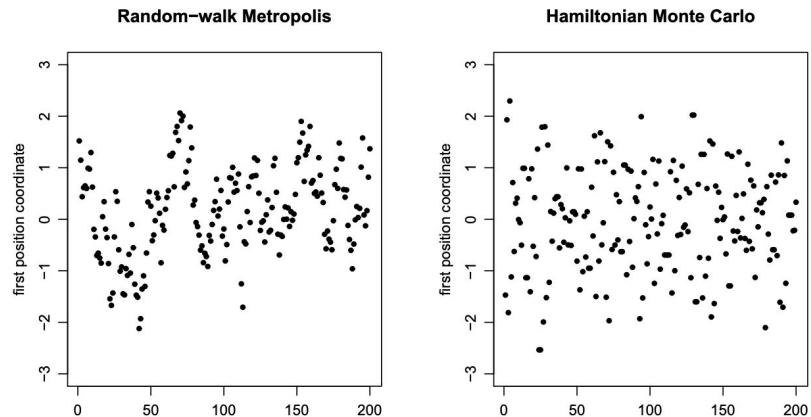
# Gradient-based MCMC methods - HMC

-Position variables ($q$) are all the variables of interest (i.e. parameters in our Bayesian model)

-Momentum variables ($p$): auxiliary variables that we introduce; same dimensionality as q

Consider system described by Hamiltonian H(p, q), where p and q are each functions of time and *Hamilton's equations* hold:

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}$$

$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}$$

These equations tell us how the system evolves in time from a starting point along *trajectories* in p,q space

Will use points along these trajectories as proposals in a Metropolis step!

Typically for HMC, assume:

$$H(q,p) = U(q) + K(p)$$

Potential energy:
negative log probability

Kinetic energy: usually use
$$K(p) = p^T M^{-1} p / 2$$
(-log p of mean-0 Gaussian)

-Position variables (**q**) are all the variables of interest (i.e. parameters in our Bayesian model)

-Momentum variables (**p**): auxiliary variables that we introduce; same dimensionality as q

Consider system described by Hamiltonian H(p, q), where p and q are each functions of time and *Hamilton's equations* hold:

Typically for HMC, assume:

$$H(q, p) = U(q) + K(p)$$

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}$$

$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}$$

Potential energy:
negative log probability

Kinetic energy: usually use

$$K(p) = p^T M^{-1} p / 2$$

(-log p of mean-0 Gaussian)

Alternate:
1.  Draw new momentum variables
2.  Metropolis updates, using Hamiltonian dynamics to propose new state (p\*, q\*)

-Position variables (**q**) are all the variables of interest (i.e. parameters in our Bayesian model)

-Momentum variables (**p**): auxiliary variables that we introduce; same dimensionality as q

Consider system described by Hamiltonian H(p, q), where p and q are each functions of time and *Hamilton's equations* hold:

Typically for HMC, assume:

$$H(q, p) = U(q) + K(p)$$

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}$$

$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}$$

Potential energy:
negative log probability

Kinetic energy: usually use

$$K(p) = p^T M^{-1} p / 2$$

(-log p of mean-0 Gaussian)

Alternate:
1. Draw new momentum variables
2. Metropolis updates, using Hamiltonian dynamics to propose new state (p*, q*)

Can show: $\frac{dH}{dt} = 0$

And Hamiltonian defines a joint probability: $P(p, q) = \frac{1}{Z} \exp(-H(q, p)/T)$

if $H(q, p) = U(q) + K(p)$

$$P(q, p) = \frac{1}{Z} \exp\left(\frac{-U(q)}{T}\right) \exp\left(\frac{-K(p)}{T}\right)$$

So, this joint probability is constant along a time trajectory

So, if we followed a trajectory exactly, our Metropolis step would accept with probability 1, since the proposed state would have same prob as initial.
In reality, we won't always accept, because we only approximate the trajectory

To compute (approx.) trajectories, need to integrate numerically. Will take $L$ steps of size $\varepsilon$. Easiest way could be Euler's method (alternate following the derivatives of p and q along a step of size $\varepsilon$:

$$p_i(t + \epsilon) = p_i(t) + \epsilon \frac{dp_i}{dt}(t) = p_i(t) - \epsilon \frac{\partial U}{\partial q_i}(q(t)), \ q_i(t + \epsilon) = q_i(t) + \epsilon \frac{q_i}{dt}(t) = q_i(t) + \epsilon \frac{p_i(t)}{m_i}$$

Euler's method **not** satisfactory here; instead use "leapfrog method" - half step momentum, full step position, another half step momentum.

$$p_i(t + \varepsilon/2) = p_i(t) - (\varepsilon/2) \frac{\partial U}{\partial q_i}(q(t)),$$

Need to choose $L$, $\varepsilon$ - they matter

$$q_i(t + \varepsilon) = q_i(t) + \varepsilon \frac{p_i(t + \varepsilon/2)}{m_i},$$

$$p_i(t + \varepsilon) = p_i(t + \varepsilon/2) - (\varepsilon/2) \frac{\partial U}{\partial q_i}(q(t + \varepsilon)).$$



(a) Euler's Method, stepsize 0.3

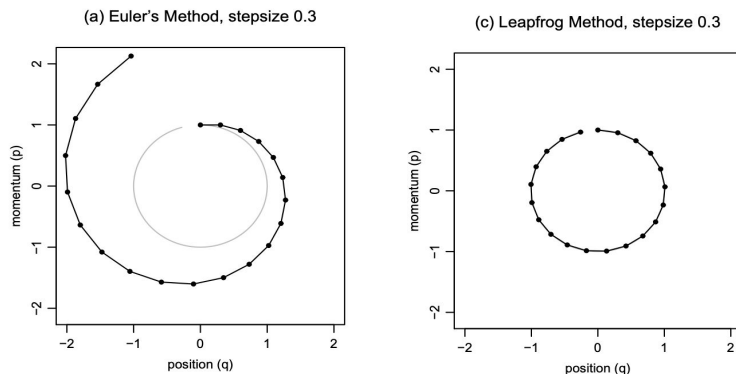(c) Leapfrog Method, stepsize 0.3

*Figure 1 from Neal, 2011*
*Position variable a zero mean gaussian*

47

# Gradient-based MCMC methods – No U-Turn Sampler

-Extension of HMC

-Big idea: choose path lengths adaptively; don't need to choose L (number of steps in the integrator)


See:

Hoffman, Matthew D., and Andrew Gelman. "The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo." J. Mach. Learn. Res. 15, no. 1 (2014): 1593-1623.

# HMC – Two excellent resources for learning more

-Neal, Radford. *MCMC using Hamiltonian dynamics*, (2011). Chapter 5 of the Handbook of Markov Chain Monte Carlo. https://arxiv.org/pdf/1206.1901.pdf

-Betancourt, Michael. *A conceptual introduction to Hamiltonian Monte Carlo*. (2018) https://arxiv.org/pdf/1701.02434.pdf

# MCMC with PyMC3

-PyMC3 is an open source Python probabilistic programming library

-Makes it easy to specify a model and fit it using algorithms like HMC/NUTS

-Integrates nicely with tools for assessing your models and results

-Documentation here: https://docs.pymc.io/ (including Quickstart: https://docs.pymc.io/pymc-examples/examples/pymc3_howto/api_quickstart.html)

-Other probabilistic programming libraries exist. E.g. Stan is very popular and an excellent choice especially if you are an R user.

*See examples in colab*

# Convergence Diagnostics - traceplots



theta
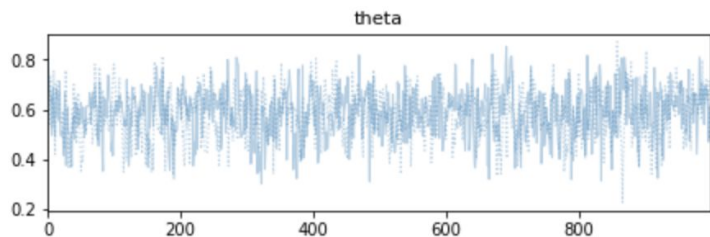
"Fuzzy caterpillar"
^ an oversimplification, but potentially a helpful one

https://canyon289.github.io/bayesian-model-evaluation/lessonplans/mcmc_basics/#/14

# Convergence Diagnostics – R-hat/Gelman-Rubin stat

-between-chain variance to within-chain variance, after running multiple chains (close to 1 is good)

# Convergence Diagnostics – effective sample size

-estimate how many independent samples (that we'd want in an ideal world) would have the same information as the dependent draws (that we have)

# Model checking

-e.g. Prior and posterior predictive checks

# More on model checking, convergence diagnostics

https://github.com/fonnesbeck/Bayes_Computing_Course/blob/master/notebooks/Section3_2-Model-Checking.ipynb

https://mc-stan.org/misc/warnings.html

Gelman et al., *Bayesian Data Analysis*, 3rd. Ed., Ch. 6, 11.4

# More on Bayesian workflow:

Gelman, A. et al., *Bayesian Workflow* (2020)
https://arxiv.org/pdf/2011.01808.pdf