Section 3.
Monte Carlo Methods

Vincent Dorie

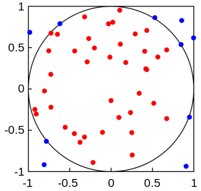
New York University

Slides courtesy of Bob Carpenter

Part I
Monte Carlo Integration

Monte Carlo Calculation of π

- Computing $\pi = 3.14\ldots$ via simulation is *the* textbook application of Monte Carlo methods.
- Generate points uniformly at random within the square
- Calculate proportion within circle ($x^2 + y^2 < 1$) and multiply by square's area (4) to produce the area of the circle.
- This area is π (radius is 1, so area is $\pi r^2 = \pi$)



Plot by Mysid Yoderj courtesy of Wikipedia.

Monte Carlo Calculation of π (cont.)

- R code to calcuate π with Monte Carlo simulation:

```
> x <- runif(1e6,-1,1)
> y <- runif(1e6,-1,1)

> prop_in_circle <- sum(x^2 + y^2 < 1) / 1e6

> 4 * prop_in_circle
[1] 3.144032
```

Accuracy of Monte Carlo

- Monte Carlo is *not* an approximation!
- It can be made exact to within any ϵ
- Monte Carlo draws are i.i.d. by definition
- Central limit theorem: expected error decreases at rate of

$$\frac{1}{\sqrt{N}}$$

- 3 decimal places of accuracy with sample size 1e6
- Need $100\times$ larger sample for each digit of accuracy

General Monte Carlo Integration

- MC can calculate arbitrary definite integrals,
$$\int_a^b f(x)\,\mathrm{d}x$$
- Let d upper bound $f(x)$ in (a, b); tightness determines computational efficiency
- Then generate random points uniformly in the rectangle bounded by (a, b) and $(0, d)$
- Multiply proportion of draws (x, y) where $y < f(x)$ by area of rectangle, $d \times (b - a)$.
- Can be generalized to multiple dimensions in obvious way

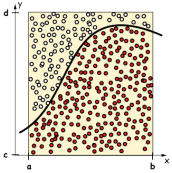


Image courtesy of Jeff Cruzan, http://www.drcruzan.com/NumericalIntegration.html

Expectations of Function of R.V.

- Suppose $f(\theta)$ is a function of random variable vector θ
- Suppose the density of θ is $p(\theta)$
- Then $f(\theta)$ is also random variable, with expectation

$$\mathbb{E}[f(\theta)] = \int f(\theta)p(\theta)\,\mathrm{d}\theta.$$

QoI as Expectations

- Most Bayesian quantities of interest (QoI) are expectations over the posterior $p(\theta \mid y)$ of functions $f(\theta)$
- **Bayesian parameter estimation**: $\hat{\theta}$
  - $f(\theta) = \theta$
  - $\hat{\theta} = \mathbb{E}[\theta \mid y]$ minimizes expected square error
- **Bayesian parameter (co)variance estimation**: $\mathrm{var}[\theta \mid y]$
  - $f(\theta) = (\theta - \mathbb{E}[\theta \mid y])^2$
- **Bayesian event probability**: $\Pr[A \mid y]$
  - $f(\theta) = \mathrm{I}[\theta \in A]$

Expectations via Monte Carlo

- Generate draws $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(M)}$ drawn from $p(\theta)$
- Monte Carlo Estimator **plugs in average** for expectation:

$$\mathbb{E}[f(\theta) \mid y] \approx \frac{1}{M} \sum_{m=1}^{M} f(\theta^{(m)})$$

- Can be made **as accurate as desired**, because

$$\mathbb{E}[f(\theta)] = \lim_{M \to \infty} \frac{1}{M} \sum_{m=1}^{M} f(\theta^{(m)})$$

- *Reminder*: By CLT, error goes down as $1 \,/\, \sqrt{M}$

Part II
Markov Chain Monte Carlo

Markov Chain Monte Carlo

- Standard Monte Carlo draws i.i.d. draws

$$\theta^{(1)}, \ldots, \theta^{(M)}$$

  according to a probability function $p(\theta)$

- Drawing an i.i.d. sample is often impossible when dealing with complex densities like Bayesian posteriors $p(\theta \mid y)$

- So we use Markov chain Monte Carlo (MCMC) in these cases and draw $\theta^{(1)}, \ldots, \theta^{(M)}$ from a Markov chain

Markov Chains

- A Markov Chain is a sequence of random variables

$$\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(M)}$$

  such that $\theta^{(m)}$ only depends on $\theta^{(m-1)}$, i.e.,

$$p(\theta^{(m)} \mid y, \theta^{(1)}, \ldots, \theta^{(m-1)}) = p(\theta^{(m)} \mid y, \theta^{(m-1)})$$

- Drawing $\theta^{(1)}, \ldots, \theta^{(M)}$ from a Markov chain according to $p(\theta^{(m)} \mid \theta^{(m-1)}, y)$ is more tractable

- Require marginal of each draw, $p(\theta^{(m)} midy)$, to be equal to true posterior

Applying MCMC

- Plug in just like ordinary (non-Markov chain) Monte Carlo
- Adjust standard errors for dependence in Markov chain

MCMC for Posterior Mean

- Standard Bayesian estimator is posterior mean

$$\hat{\theta} \;=\; \int \theta \, p(\theta|y) \, \mathrm{d}\theta$$

  - Posterior mean minimizes expected square error

- Estimate is a conditional expectation

$$\hat{\theta} = \mathbb{E}[\theta \mid y]$$

- Compute by averaging

$$\hat{\theta} \;\approx\; \frac{1}{M} \sum_{m=1}^{M} \theta^{(m)}$$

MCMC for Posterior Variance

- Posterior variance works the same way,

$$\mathbb{E}[(\theta - \mathbb{E}[\theta \mid y])^2 \mid y] \approx \frac{1}{M} \sum_{m=1}^{M} (\theta^{(m)} - \hat{\theta})^2$$

MCMC for Event Probability

- Event probabilities are also expectations, e.g.,

$$\Pr[\theta_1 > \theta_2] = \mathbb{E}[\mathrm{I}[\theta_1 > \theta_2]] = \int_{\Theta} \mathrm{I}[\theta_1 > \theta_2]\, p(\theta \mid y) \mathrm{d}\theta.$$

- Estimation via MCMC just another plug-in:

$$\Pr[\theta_1 > \theta_2] \approx \frac{1}{M} \sum_{m=1}^{M} \mathrm{I}[\theta_1^{(m)} > \theta_2^{(m)}]$$

- Again, can be made as accurate as necessary

MCMC for Quantiles (incl. median)

- These are not expectations, but still plug in
- Alternative Bayesian estimator is posterior median
  - Posterior median minimizes expected absolute error
- Estimate as median draw of $\theta^{(1)}, \ldots, \theta^{(M)}$
  - just sort and take halfway value
  - e.g., Stan shows 50% point (or other quantiles)
- Other quantiles including interval bounds similar
  - estimate with quantile of draws
  - estimation error goes up in tail (based on fewer draws)

Part III
MCMC Algorithms

Random-Walk Metropolis

- We want to sample from a density $p(\theta)$
- Take current state, add random pertubation
  - Proposal $\theta^* = \theta^{(m)} + \epsilon$
- Accept new θ with a probability that gives samples desired probability

  Demo

- **Proposal** distribution determines behavior of chain

Random-Walk Formal Def

- Draw random initial parameter vector $\theta^{(1)}$ (in support)
- For $m = 2, \ldots, M$
  - Sample **proposal** from a (symmetric) jumping distribution, e.g.,

    $$\theta^* \sim \text{MultiNormal}(\theta^{(m-1)}, \sigma \mathrm{I})$$

    where I is the identity matrix
  - Draw $u \sim \text{Uniform}(0, 1)$ and set

    $$\theta^{(m)} = \begin{cases} \theta^* & \text{if } u < \dfrac{p(\theta^* \mid y)}{p(\theta^{(m-1)} \mid y)} \\ \theta^{(m-1)} & \text{otherwise} \end{cases}$$

Metropolis-Hastings

- Generalizes Metropolis to asymmetric proposals (i.e. $J(\theta^* \mid \theta^{(m)}) \neq J(\theta^{(m)} \mid \theta^*)$)

- Acceptance ratio is

$$\frac{J(\theta^{(m)} \mid \theta^*) \ \times \ p(\theta^* \mid y)}{J(\theta^* \mid \theta^{(m-1)}) \ \times \ p(\theta^{(m)} \mid y)}$$

  where J is the proposal density

- i.e.,

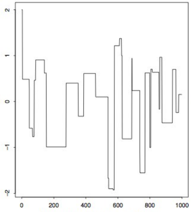
$$\frac{\text{probabilty of being at } \theta^* \text{ and jumping to } \theta^{(m-1)}}{\text{probability of being at } \theta^{(m-1)} \text{ and jumping to } \theta^*}$$

Metropolis-Hastings (cont.)

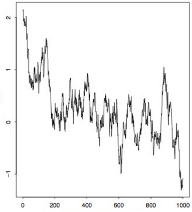
- General form ensures equilibrium by maintaining *detailed balance*
- Like Metropolis, only requires ratios
- Many algorithms involve a Metropolis-Hastings “correction”
  - Including vanilla HMC and RHMC and ensemble samplers
- **Detailed Balance** & **Reversibility** formal conditions to guarantee *convergence* of chain to desired distribution

Optimal Proposal Scale?

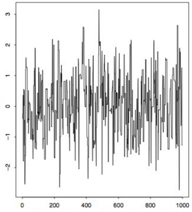
- Proposal scale σ is a free; too low or high is inefficient



(a) Proposal variance too large



(b) Proposal variance too small



(c) Proposal variance approximately optimised

- *Traceplots* show parameter value on y axis, iterations on x
- Empirical tuning problem; theoretical optima exist for some cases

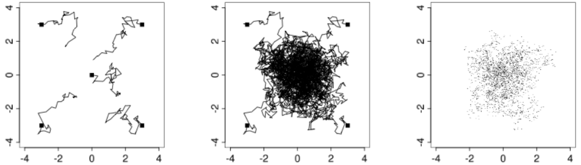
Roberts and Rosenthal (2001) Optimal Scaling for Various Metropolis-Hastings Algorithms. *Statistical Science*.

Convergence

- Markov chains depend on previous state
- Construct so that stationary (long-run) state is target
- What about starting state?
- Imagine a signal problem on the MTA
  - A trains on the F line to Brooklyn, C service suspended...
  - Substantial delays at W 4th St
  - Eventually... service returns to normal
- May take many iterations for Markov chain to reach equilibrium

Convergence: Example

- Test for convergence: start multiple Markov chains at diffuse points



- Four chains with different starting points
  - *Left*: 50 iterations
  - *Center*: 1000 iterations
  - *Right*: Draws from second half of each chain

Gelman et al., *Bayesian Data Analysis*

Convergence Diagnostic ($\hat{R}$)

- Gelman & Rubin recommend M chains of N draws with **diffuse initializations**
- Measure that each chain has same posterior mean and variance
- If not, may be stuck in multiple modes or just not converged yet
- Define statistic $\hat{R}$ of chains s.t. **at convergence**, $\hat{R} \to 1$
  - $\hat{R} >> 1$ implies non-convergence
  - $\hat{R} \approx 1$ **does not guarantee convergence**
  - Only measures marginals

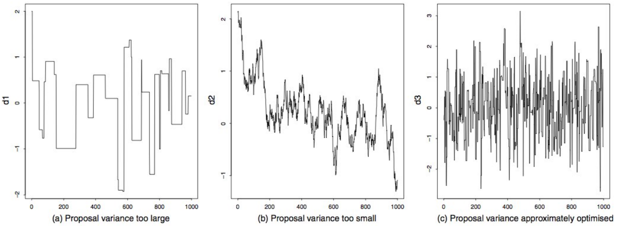
Effective Sample Size

- Markov chains typically display autocorrelation in the series of draws $\theta^{(1)}, \ldots, \theta^{(m)}$
- Without i.i.d. draws, central limit theorem *does not apply*
- Effective sample size N_{eff} divides out autocorrelation
- N_{eff} must be estimated from sample
- Estimation accuracy proportional to

$$\frac{1}{\sqrt{N_{\text{eff}}}}$$

Reducing Posterior Correlation

- Tuning algorithm parameters to ensure good mixing
- Recall Metropolis traceplots of Roberts and Rosenthal:



(a) Proposal variance too large (b) Proposal variance too small (c) Proposal variance approximately optimised

- Good jump scale σ produces good mixing and high N_{eff}

Gibbs Sampling

- Draw random initial parameter vector $\theta^{(1)}$ (in support)
- For $m = 2, \ldots, M$:
  - For $n = 1, \ldots, N$:
    * draw $\theta_n^{(m)}$ according to conditional

      $$p(\theta_n \mid \theta_1^{(m)}, \ldots, \theta_{n-1}^{(m)}, \theta_{n+1}^{(m-1)}, \ldots, \theta_N^{(m-1)}, y).$$
- e.g, with $\theta = (\theta_1, \theta_2, \theta_3)$:
  - draw $\theta_1^{(m)}$ according to $p(\theta_1 \mid \theta_2^{(m-1)}, \theta_3^{(m-1)}, y)$
  - draw $\theta_2^{(m)}$ according to $p(\theta_2 \mid \theta_1^{(m)}, \theta_3^{(m-1)}, y)$
  - draw $\theta_3^{(m)}$ according to $p(\theta_3 \mid \theta_1^{(m)}, \theta_2^{(m)}, y)$

Generalized Gibbs

- “Proper” Gibbs requires conditional Monte Carlo draws
  - typically works only for conjugate priors
- In general case, may need to use less efficient conditional draws
  - Slice sampling is a popular general technique that works for discrete or continuous θ_n (JAGS)
  - Adaptive rejection sampling is another alternative (BUGS)
  - Very difficult in more than one or two dimensions

Sampling Efficiency

- We care only about N_{eff} per second
- Decompose into
  1. Iterations per second
  2. Effective sample size per iteration
- Gibbs and Metropolis have high iterations per second (especially Metropolis)
- But they have low effective sample size per iteration (especially Metropolis)
- Both are particular weak when there is high correlation among the parameters in the posterior

Hamiltonian Monte Carlo & NUTS

- Slower iterations per second than Gibbs or Metropolis
- Much higher effective sample size per iteration for complex posteriors (i.e., high curvature and correlation)
- Overall, much higher N_{eff} per second

- Details in the next talk . . .
- Along with details of how Stan implements HMC and NUTS

Why Stan is Great

- Writing a sampler for model:

$$
\begin{aligned}
y \mid \beta, \sigma^2 &\sim \text{Normal}(x\beta, \sigma_y^2),\\
\beta &\sim \text{Normal}(0, \sigma_\beta^2),\\
\sigma &\sim \text{Half-Cauchy}.
\end{aligned}
$$

The End (Section 3)

Part I
What Stan Does

Full Bayes: No-U-Turn Sampler

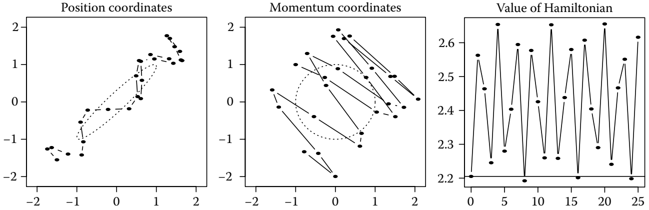
- Adaptive **Hamiltonian Monte Carlo** (HMC)
  - **Potential Energy**: negative log posterior
  - **Kinetic Energy**: random standard normal per iteration
- Adaptation **during warmup**
  - step size adapted to target total acceptance rate
  - mass matrix estimated with regularization
- Adaptation **during sampling**
  - simulate forward and backward in time until U-turn
- **Slice sample** along path

(Hoffman and Gelman 2011, 2014)

Sample: Hamiltonian Flow

- Generate random **kinetic energy**
  - random Normal$(0, 1)$ in each parameter
- Use negative log posterior as **potential energy**
- Hamiltonian is kinetic plus potential energy
- **Leapfrog Integration**: for *fixed* stepsize (time discretization), number of steps (total time), and mass matrix,
  - update momentum half-step based on potential (gradient)
  - update position full step based on momentum
  - update momentum half-step based on potential
- Numerical solution of Hamilton's first-order version of Newton's second-order diff-eqs of motion (force $=$ mass $\times$ acceleration)

Sample: Leapfrog Example



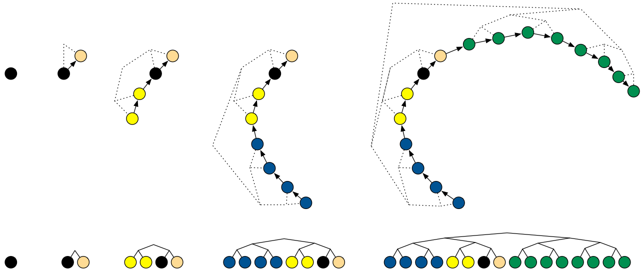
- Trajectory of 25 leapfrog steps for correlated 2D normal (ellipses at 1 sd from mean), stepsize of 0.25, initial state of $(-1, 1)$, and initial momentum of $(-1.5, -1.55)$.

Radford Neal (2013) MCMC using Hamiltonian Dynamics. In *Handbook of MCMC*. (free online at http://www.mcmchandbook.net/index.html)

Sample: No-U-Turn Sampler (NUTS)

- Adapts Hamiltonian simulation time
  - goal to maximize mixing, maintaining detailed balance
  - too short devolves to random walk
  - too long does extra work (i.e., orbits)
- For exponentially increasing number of steps up to max
  - Randomly choose to extend forward or backward in time
  - Move forward or backward in time number of steps
    - stop if any subtree (size 2, 4, 8, ...) makes U-turn
    - remove all current steps if subtree U-turns (not ends)
- Randomly select param with density above slice (or reject)

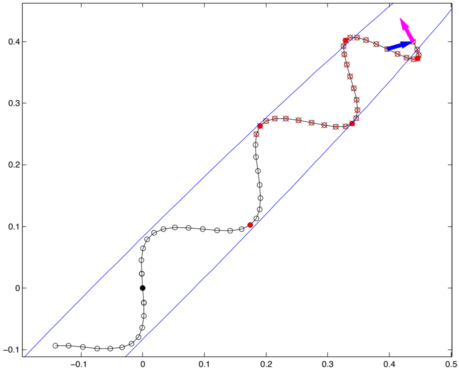
Sample: NUTS Binary Tree



- Example of repeated doubling building binary tree forward and backward in time until U-turn.

  Hoffman and Gelman. 2014. The No-U-Turn Sampler. *JMLR*. (free online at http://jmlr.org/papers/v15/hoffman14a.html)

Sample: NUTS U-Turn



- Example of trajectory from one iteration of NUTS.
- Blue ellipse is contour of 2D normal.
- Black circles are leapfrog steps.
- Solid red circles excluded below slice
- U-turn made with blue and magenta arrows
- Red crossed circles excluded for detailed balance

Sample: HMC/NUTS Warmup

- Estimate stepsize
  - too small requires too many leapfrog steps
  - too large induces numerical inaccuracy
  - need to balance
- Estimate mass matrix
  - Diagonal accounts for parameter scales
  - Dense optionally accounts for rotation

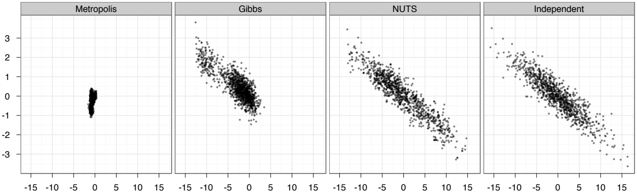
Sample: Warmup (cont.)

- Initialize unconstrained parameters as for optimization
- For exponentially increasing block sizes
  - for each iteration in block
    - generate random kinetic energy
    - simulate Hamiltonian flow (HMC fixed time, NUTS adapts)
    - choose next state (Metroplis for HMC, slice for NUTS)
  - update regularized point estimate of mass matrix
    - use parameter draws from current block
    - shrink diagonal toward unit; dense toward diagonal
  - tune stepsize (line search) for target acceptance rate

Sample: HMC/NUTS Sampling

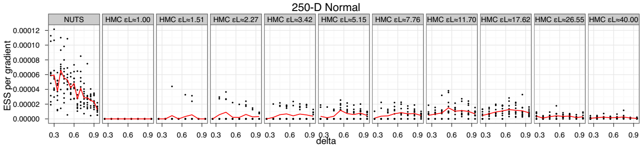
- Fix stepsize and and mass matrix
- For sampling iterations
    - generate random kinetic energy
    - simulate Hamiltonian flow
    - apply Metropolis accept/reject (HMC) or slice (NUTS)

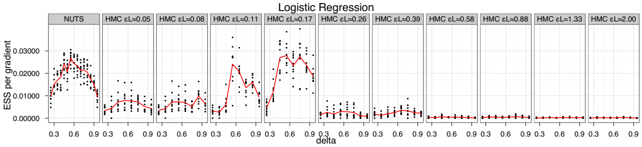
NUTS vs. Gibbs and Metropolis



- Two dimensions of highly correlated 250-dim normal
- **1,000,000 draws** from Metropolis and Gibbs (thin to 1000)
- **1000 draws** from NUTS; 1000 independent draws

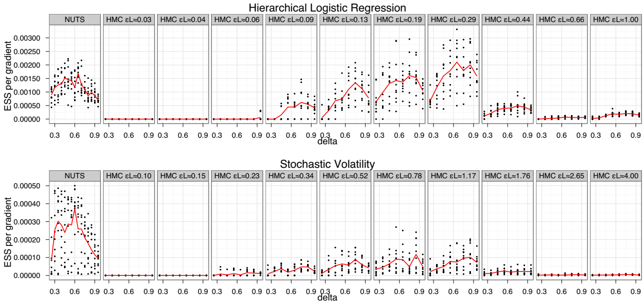
NUTS vs. Basic HMC





- 250-D normal and logistic regression models
- Vertical axis is effective sample size per sample (bigger better)
- Left) NUTS; Right) HMC with increasing $t = \epsilon L$

NUTS vs. Basic HMC II



- Hierarchical logistic regression and stochastic volatility
- Simulation time t is $\epsilon\, L$, step size (ϵ) times number of steps (L)
- NUTS can beat optimally tuned HMC (latter very expensive)

Part III

Under Stan’s Hood

Euclidean Hamiltonian

- **Phase space**: q position (parameters); p momentum
- **Posterior density**: $\pi(q)$
- **Mass matrix**: M
- **Potential energy**: $V(q) = -\log \pi(q)$
- **Kinetic energy**: $T(p) = \frac{1}{2} p^\top M^{-1} p$
- **Hamiltonian**: $H(p, q) = V(q) + T(p)$
- **Diff eqs**:

$$\frac{dq}{dt} = +\frac{\partial H}{\partial p} \qquad \qquad \frac{dp}{dt} = -\frac{\partial H}{\partial q}$$

Leapfrog Integrator Steps

- Solves Hamilton's equations by **simulating dynamics**
  (symplectic [volume preserving]; ϵ^3 error per step, ϵ^2 total error)
- Given: **step size** ϵ, **mass matrix** M, **parameters** q
- **Initialize kinetic** energy, $p \sim \text{Normal}(0, \mathbf{I})$
- **Repeat** for L leapfrog steps:

$$p \leftarrow p - \frac{\epsilon}{2} \frac{\partial V(q)}{\partial q} \qquad \text{[half step in momentum]}$$

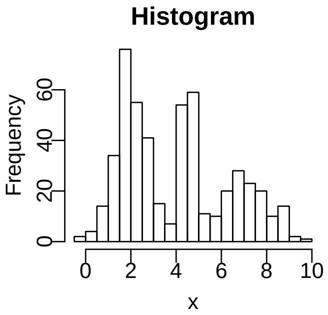
$$q \leftarrow q + \epsilon M^{-1} p \qquad \text{[full step in position]}$$

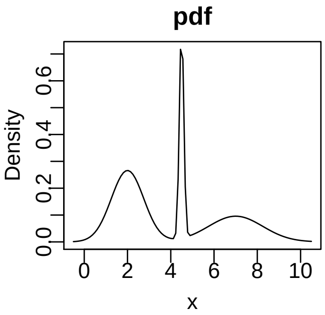
$$p \leftarrow p - \frac{\epsilon}{2} \frac{\partial V(q)}{\partial q} \qquad \text{[half step in momentum]}$$

Part IV
Mixutre Models

Mixture Models

- Class of models with observations coming from than one distribution, membership unknown





Mixture Model Density

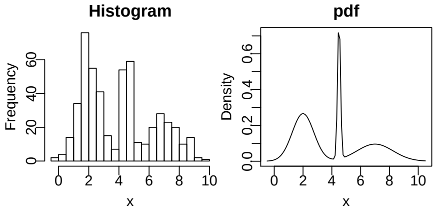
- Depends on a **latent** class membership

$$y_i \mid z_i \sim \begin{cases} p_1(y_i) & \text{if } z_i = 1 \\ p_2(y_i) & \text{if } z_i = 2 \\ \vdots & \\ p_K(y_i) & \text{if } z_i = K \end{cases},$$

$$z_i \sim \text{Categorical}(\lambda).$$

$\lambda \geq 0, \sum_{k=1}^{K} \lambda_k = 1$

Example Density



- $y_i \mid z_i \sim$ Normal(4.5, 0.1) if $z_i = 1$,
- $y_i \mid z_i \sim$ Normal(2, 2) if $z_i = 2$,
- $y_i \mid z_i \sim$ Normal(7, 1.25) if $z_i = 3$,
- $\lambda = (0.2, 0.5, 0.3)$

```
data {
  int<lower = 0> N;
  int<lower = 0> K;
  real y[N];
}
parameters {
  simplex[K] lambda;
  int z[K];

  real[K] mu;
  real<lower = 0> sigma[K];
}
model {
  z ~ categorical(lambda);
  for (n in 1:n)
    y[n] ~ normal(mu[z[n]], sigma[z[n]])
}
```

```
data {
  int<lower = 0> N;
  int<lower = 0> K;
  real y[N];
}
parameters {
  simplex[K] lambda;
  int z[K];

  real[K] mu;
  real<lower = 0> sigma[K];
}
model {
  z ~ categorical(lambda);
  for (n in 1:n)
    y[n] ~ normal(mu[z[n]], sigma[z[n]])
}
```

```
integer parameters or transformed parameters are not allowed;
found declared type int, parameter name=z Problem with
declaration.
```

Integrate/Sum Out z

$$p(y_i, z_i) = [\mathbb{I}[z_i = 1]p_1(y_i) + \mathbb{I}[z_i = 2]p_2(y_i) + \cdots + \mathbb{I}[z_i = K]p_K(y_i)] \prod_{k=1}^{K} \lambda_k^{\mathbb{I}[z_i=k]}.$$

$$\sum_{z_i=1}^{K} p(y_i, z_i) = \lambda_1 p_1(y_i) + \lambda_2 p_2(y_i) + \cdots + \lambda_K p_K(y_i),$$

$$= p(y_i).$$

Multiply together to get likelihood:

$$p(y) = \prod_{i=1}^{N} [\lambda_1 p_1(y_i) + \lambda_2 p_2(y_i) + \cdots + \lambda_K p_K(y_i)].$$

`log` Problem

- Operate on log probability as multiplication leads to underflow
  - `small #` $\times$ `small #` $\times$ ...
  - $\log($`small #`$) + \log($`small #`$)+$ ...
- Log-likelihood is unwieldy:

$$\log p(y) = \sum_{i=1}^{N} \log \left[\lambda_1 p_1(y_i) + \lambda_2 p_2(y_i) + \cdots + \lambda_K p_K(y_i)\right].$$

`log_sum_exp`