Statistical Inference

Lecture 12a

ANU - RSFAS

Last Updated: Sun May 20 16:06:52 2018

Generating Random Variables - Inverse CDF Method

- Consider $X \sim \text{exponential}(\beta = 2)$:

$$F_X(c) = \int_0^c \frac{1}{\beta} exp(-x/\beta)dx = 1 - exp(-c/\beta)$$

$$U = F_X(X) = 1 - exp(-X/\beta)$$

$$\begin{aligned} U &= F_X(X) = 1 - exp(-X/\beta) \\ 1 - U &= exp(-X/\beta) \\ log(1-U) &= -X/\beta \\ -\beta log(1-U) &= X = F_X^{-1}(U) \end{aligned}$$

Generating Random Samples

- For an exponential (β) distribution we have:

$$Y_i = -\beta log(1 - U_i)$$

As U is uniform $(0, 1)$ then we can simply sample by:

$$Y_i = -\beta log(U_i)$$

- Let's prove that if $U \sim \mathrm{uniform}(0, 1)$ then $Y = 1 - U \sim \mathrm{uniform}(0, 1)$.

Generating Random Samples

- Based on the uniform-exponential relationship we can generate the following:
  - Sums of iid exponential random variables have a gamma distribution:

$$Y = -\beta \sum_{j=1}^{a} log(U_j) \sim \mathrm{gamma}(a, \beta)$$

  - If $\beta = 2$, then the distribution is a χ^2 random variable:

$$Y = -2 \sum_{j=1}^{v} log(U_j) \sim \chi^2_{2v}$$

  - The ratio of sums of exponentials is a beta distribution:

$$Y = \frac{\sum_{j=1}^{a} log(U_j)}{\sum_{j=1}^{a+b} log(U_j)} \sim \mathrm{beta}(a, b)$$

Generating Random Samples

- Let's generate some beta $(a = 2, b = 5)$ random variables.
- If $X \sim \mathrm{beta}(a = 2, b = 5)$, then

$$E[X] = \frac{a}{a+b} = \frac{2}{2+5} = 0.2857$$

$$V[X] = \frac{ab}{(a+b)^2(a+b+1)} = \frac{2(5)}{(2+5)^2(2+5+1)} = 0.02551$$

```r
set.seed(1001)
n <- 10000
a <- 2
b <- 5
y <- rep(0, n)
for(i in 1:n){
  u <- runif(a+b, 0, 1)
  y[i] <- sum(log(u[1:a]))/sum(log(u[1:(a+b)]))
  }

mean(y)
```

$p(\theta|y) \propto p(y|\theta)\, p(\theta)$

```
## [1] 0.2853618
```

```r
var(y)
```

```
## [1] 0.02523963
```

Generating Random Samples

- Examine the following again:
  - If $\beta = 2$, then the distribution is a χ^2 random variable:

$$Y = -2\sum_{j=1}^{v} log(U_j) \sim \chi^2_{2v}$$

This suggests that we cannot simulate a χ^2_1 (or an odd number for v) random variable with this approach!

can do χ^2_2, χ^2_4, χ^2_6, ...

- If we could generate a normal$(0, 1)$ then we could generate a χ^2_1.
- There is no closed form solution to generate a single normal$(0, 1)$.
- Surprisingly through we can generate two independent normal$(0, 1)$ random variables!

Generating Random Samples

- Example (Box-Muller Algorithm):
  - Generate $U_1,\ U_2 \sim \mathrm{uniform}(0,1)$.
  - Set:

$$R = \sqrt{-2log(U_1)}, \quad \theta = 2\pi U_2$$

  - Then:

$$X = Rcos(\theta), \qquad Y = Rsin(\theta)$$

  - Then $X, Y \overset{\mathrm{iid}}{\sim} \mathrm{normal}(0,1)$
- If we want two samples from a χ^2_1 all we have to do is:

$$X^2, Y^2$$

Generating Random Samples

- So far we have considered continuous distributions.

$$F_Y^{-1}(u) = y \leftrightarrow u = \int_{-\infty}^{y} f_Y(t)dt$$

- Now let's sample from discrete distributions.
- If Y is a discrete random variable taking on values:

$$y_1 < y_2 < \cdots < y_k$$

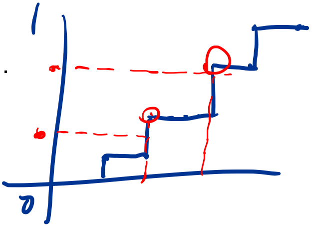
then we can write:

$$\begin{aligned} P[F_Y(y_i) < U \leq F_Y(y_{i+1})] &= F_Y(y_{i+1}) - F_Y(y_i) \\ &= P(Y = y_{i+1}) \end{aligned}$$

Generating Random Samples

Using this idea we can easily discrete random variables. To generate $Y_i \sim f_Y(y)$:

1. Generate $U \sim \mathrm{uniform}(0,1)$.
2. If $F_Y(y) < U \leq F_Y(y_{i+1}), set Y = y_{i+1}$.



Define $y_0 = -\infty$ and $F_Y(y_0) = 0$.

Generating Random Samples

- Example (Binomial random variable generation)
- Let's generate random variables from $Y \sim \mathrm{binomial}(n = 4, p = 5/8)$.

CDF

1. Generate $U \sim \mathrm{uniform}(0, 1)$.
2. Determine Y:

$$Y = \begin{cases} 0 & \text{if } 0 < U \leq 0.020 \\ 1 & \text{if } 0.020 < U \leq 0.152 \\ 2 & \text{if } 0.152 < U \leq 0.481 \\ 3 & \text{if } 0.481 < U \leq 0.847 \\ 4 & \text{if } 0.847 < U \leq 1 \end{cases}$$

Generating Random Samples

```
set.seed(2001)
n <- 10000

u <- runif(n, 0,1)
y <- qbinom(u, 4, 5/8)

mean(y)
```

$n \cdot p$

```
## [1] 2.4971
```

```
var(y)
```

$np(1-p)$

```
## [1] 0.9496866
```

$E[Y] = np = 4(5/8) = 2.5, \quad V[Y] = np(1-p) = 4(5/8)(1-5/8) = 0.9375$

Indirect Sampling Methods

- Thus we we considered direct sampling methods (generate X then apply a function to get Y directly), now we will consider indirect methods.
- Indirect methods are useful when we don't have a nice analytical solution to the inverse of the function of interest.

Generating Random Samples

Theorem (The Accept/Reject Algorithm):

- Let $Y \sim f_Y(y)$ and $V \sim f_V(v)$, where densities have common support and

$$M = \overset{\text{max}}{\underset{y}{sup}} \frac{f_Y(y)}{f_V(y)} < \infty$$

- Suppose we want to sample from Y and are able to sample from V.

1. Generate $U \sim \mathrm{uniform}(0, 1)$ and $V \sim f_V$, independently.
2. If $U < \frac{1}{M}\frac{f_Y(V)}{f_v(V)}$, set $Y = V$; otherwise return to (1).

Note: envelope $= M\ f_V(v) \geq f_Y(v)$.

Generating Random Samples

- **Example**:
  - We know how to generate $V \sim \text{beta}(2, 6)$, see slide ~~3~~. 4
  - Now let's generate $Y \sim \text{beta}(2.7, 6.3)$. The previous method will not work!
  - Lets first figure out M:

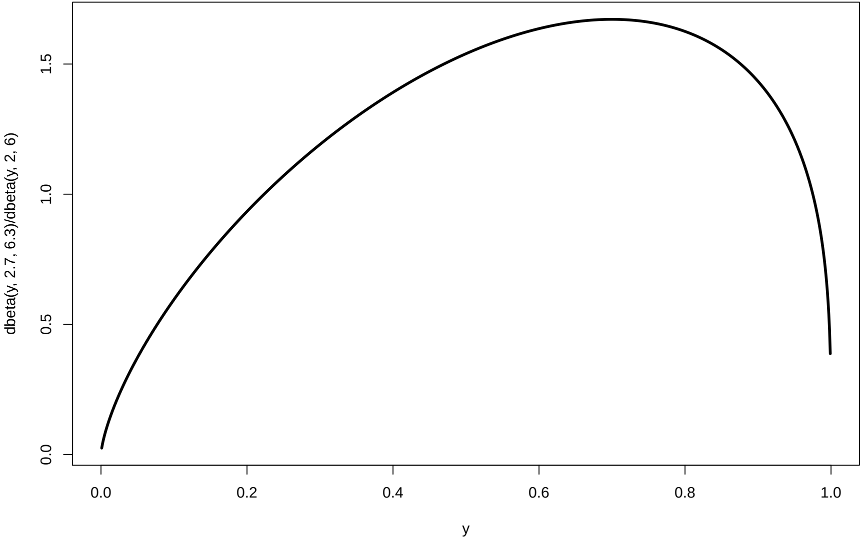
$$M = \overset{\text{max}}{sup_y} \frac{f_Y(y)}{f_V(y)}$$

```r
y <- seq(0.001, 0.999, by=0.001)
M <- max(dbeta(y, 2.7, 6.3)/dbeta(y, 2, 6))
M
```

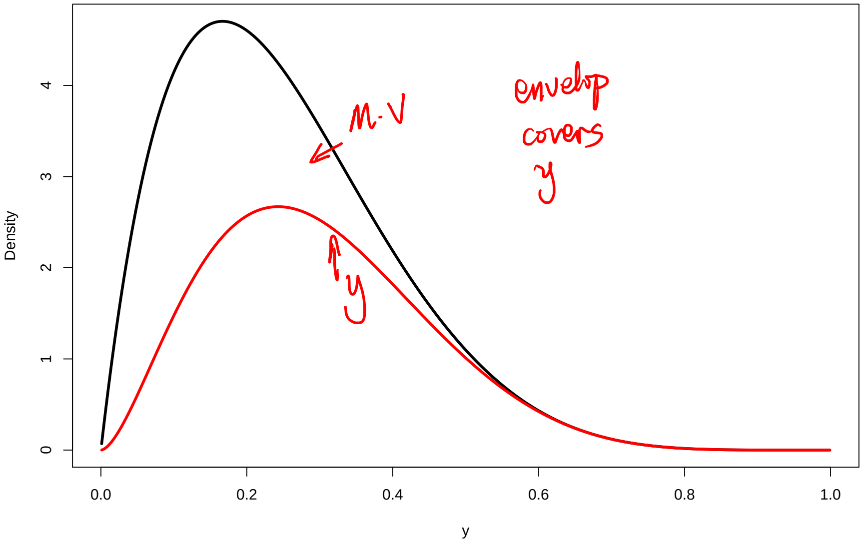
f_y / f_v

```
## [1] 1.671808
```

```r
plot(y, dbeta(y, 2.7, 6.3)/dbeta(y, 2, 6), type="l", lwd=3)
```



```r
plot(y, M*dbeta(y, 2, 6), type="l", lwd=3, ylab="Density")
lines(y, dbeta(y, 2.7, 6.3), lwd=3, col="red")
```



Generating Random Samples

```
set.seed(1001)
n <- 10000
y <- NULL

for(i in 1:n){
u <- runif(1, 0, 1)
v <- rbeta(1, 2, 6)
if(u < (1/M)*(dbeta(v, 2.7, 6.3)/dbeta(v, 2, 6))){
 y.i <- v
 y <- c(y, y.i)
  }}
length(y)
```

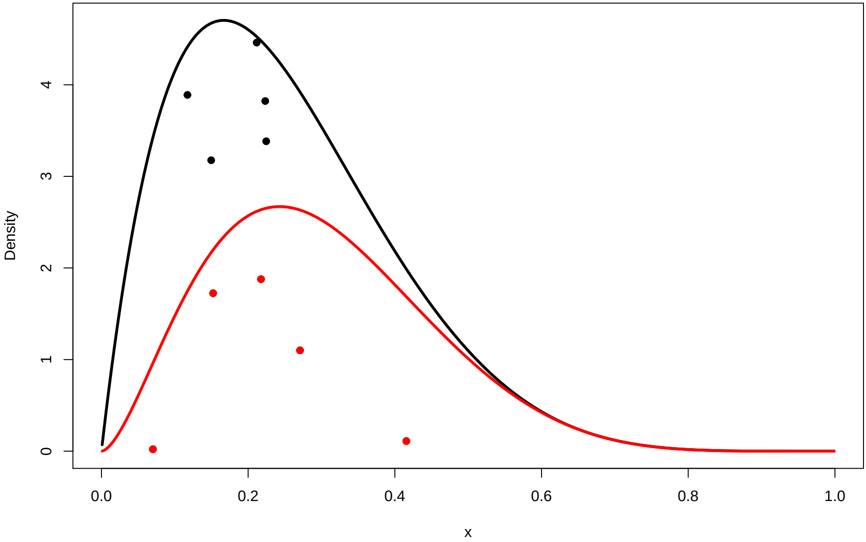
```
## [1] 6039
```

First 10 Draws

```
set.seed(1001)
n <- 10

m.v.u <- rep(0, 10)
v.out <- rep(0, 10)
y.out <- rep(0,10)


for(i in 1:n){
u <- runif(1, 0, 1)
v <- rbeta(1, 2, 6)

v.out[i] <- v
m.v.u[i] <- M*dbeta(v, 2, 6)*u

if(u < (1/M)*(dbeta(v, 2.7, 6.3)/dbeta(v, 2, 6))){

 y.out[i] <- 1

  }}
```

```
x <- seq(0.001, 0.999, by=0.001)
plot(x, M*dbeta(x, 2, 6), type="l", lwd=3, ylab="Density")
lines(x, dbeta(x, 2.7, 6.3), lwd=3, col="red")
points(v.out, m.v.u, pch=19)
points(v.out[y.out==1], m.v.u[y.out==1], pch=19, col="red")
```

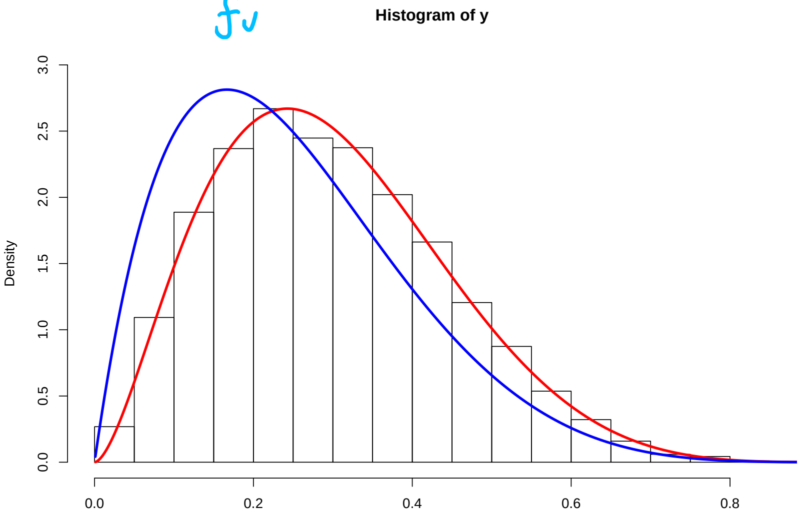
First 10 Draws



```
hist(y, prob=TRUE, ylim=c(0,3))
x <- seq(0.001, 0.999, by=0.001)
lines(x, dbeta(x, 2.7, 6.3), lwd=3, col="red")
lines(x, dbeta(x, 2, 6), lwd=3, col="blue")
```

hist

fu



Proof:

$$
\begin{aligned}
P(Y \leq y) &= P\left(V \leq y | \text{select}\right) \\
&= P\left(\overbrace{V \leq y}^{A} \Big| \overbrace{U < \frac{1}{M}\frac{f_Y(V)}{f_V(V)}}^{B}\right) \\
&= \frac{P\left(V \leq y \text{ and } U < \frac{1}{M}\frac{f_Y(V)}{f_V(V)}\right)}{P\left(U < \frac{1}{M}\frac{f_Y(V)}{f_V(V)}\right)} \\
&= \frac{\int_{-\infty}^{y}\int_{0}^{\frac{1}{M}\frac{f_Y(v)}{f_V(v)}} f_U(u) f_V(v)\,du\,dv}{\int_{-\infty}^{\infty}\int_{0}^{\frac{1}{M}\frac{f_Y(v)}{f_V(v)}} f_U(u) f_V(v)\,du\,dv} = \frac{\int_{-\infty}^{y}\int_{0}^{\frac{1}{M}\frac{f_Y(v)}{f_V(v)}} 1 f_V(v)\,du\,dv}{\int_{-\infty}^{\infty}\int_{0}^{\frac{1}{M}\frac{f_Y(v)}{f_V(v)}} 1 f_V(v)\,du\,dv} \\
&= \frac{\int_{-\infty}^{y}\frac{1}{M} f_Y(v)\,dv}{\frac{1}{M}} = \int_{-\infty}^{y} f_Y(v)\,dv
\end{aligned}
$$

$$\frac{P(A\&B)}{P(B)}$$

Generating Random Samples

- What can we say about M?

$$
\begin{aligned}
P\left(\overset{\text{select}}{\cancel{\text{stop}}}\right) &= P\left(U < \frac{1}{M}\frac{f_Y(V)}{f_V(V)}\right) \\
&= \int_{-\infty}^{\infty}\int_0^{\frac{1}{M}\frac{f_Y(v)}{f_V(v)}} f_U(u) f_V(v)\ du\ dv = \int_{-\infty}^{\infty}\int_0^{\frac{1}{M}\frac{f_Y(v)}{f_V(v)}} 1\ du\ f_V(v) dv \\
&= \int_{-\infty}^{\infty} \frac{1}{M}\frac{f_Y(v)}{\cancel{f_V(v)}} \cancel{f_V(v)} dv \\
&= \int_{-\infty}^{\infty} \frac{1}{M} f_Y(v) dv \\
&= \frac{1}{M}\int_{-\infty}^{\infty} f_Y(v) dv \\
&= \frac{1}{M} \times \underset{\sim}{1} = \frac{1}{M}
\end{aligned}
$$

Generating Random Samples

- We are considering the number of trials till a success (a geometric distribution). If $W \sim \text{geometric}(\theta)$ then $E[W] = 1/\theta$.
  - The probability of success is:

$$p = 1/M$$

  - The expected number of draws till a success:

$$1/p = M$$

  - In our example we found $M = 1.672$. In the end we had $6,039$ successes.

$$6,039 \times 1.672 = 10,097.21 \approx n = 10,000$$

Generating Random Samples

- Various specialized versions of this technique exist to solve particular problems (See Givens and Hoeting):
  - Squeezed Rejection Sampling (cases where evaluating $f_Y(y)$ is computationally expensive)
  - Adaptive Rejection Sampling (adaptively generates a suitable envelope).

Generating Random Samples

- For the standard accept/reject algorithm we need a good envelope. For some distributions that may be difficult.
- When a good envelope is not available Markov chain Monte Carlo (MCMC) can aid in sampling for a desired target distribution:
  - Metropolis algorithm
  - Metropolis-Hastings algorithm
  - Gibbs sampling
  - ...

Metropolis-Hastings Algorithm

jumping distribution

- Let $Y \sim f_Y(y)$ and $Y^* \sim f_V(v)$, where f_Y and f_V have common support. Then to generate $Y \sim f_Y$:
  1. Set $Z_0 = c$ any starting value. This could be by drawing a Y^* from $f_V(v)$.
  2. For $i = 1, \ldots$:
     2.1 Generate $Y_i^* \sim f_V$ and $U_i \sim \mathrm{uniform}(0,1)$ and calculate:

$$\rho_i = \min\left\{ \underbrace{\frac{f_Y(Y_i^*)}{f_Y(Z_{i-1})}}_{\text{ratio of target density}} \times \underbrace{\frac{f_V(Z_{i-1})}{f_V(Y_i^*)}}_{\text{ratio of proposal density}}, 1 \right\}$$

     2.2 Set

$$Z_i = \begin{cases} Y_i^* & \text{if } U_i \le \rho_i \\ Z_{i-1} & \text{if } U_i > \rho_i \end{cases}$$

As $i \to \infty$, Z_i converges to Y in distribution.

Metropolis Algorithm

- If the proposal distribution is symmetric, $f_V(Z_{i-1}|Y_i^*) = f_V(Y_i^*|Z_{i-1})$, then we have the Metropolis algorithm:

  1. Set $Z_0 = c$ any starting value. This could be by drawing a Y^* from $f_V(v)$.
  2. For $i = 1, \ldots$:

     2.1 Generate $Y_i^* \sim f_V$ and $U_i \sim \mathrm{uniform}(0,1)$ and calculate:

$$\rho_i = \min\left\{\frac{f_Y(Y_i^*)}{f_Y(Z_{i-1})}, 1\right\}$$

     2.2 Set

$$Z_i = \begin{cases} Y_i^* & \text{if } U_i \le \rho_i \\ Z_{i-1} & \text{if } U_i > \rho_i \end{cases}$$

As $i \to \infty$, Z_i converges to Y in distribution.

Metropolis Algorithm

- Intuition:
  - If $\frac{f_Y(Y_i^*)}{f_Y(Z_{i-1})} > 1$, then accept Y^* as it has a higher 'probability' than Z_{i-1}.
  - If $r = \frac{f_Y(Y_i^*)}{f_Y(Z_{i-1})} \leq 1$, then accept Y^* at the rate r.
- Common symmetric proposal distributions:
  - $f_V(Y_i^*|Z_{i-1}) = \mathrm{uniform}(Z_{i-1} - \delta, Z_{i-1} + \delta)$
  - $f_V(Y_i^*|Z_{i-1}) = \mathrm{normal}(\mu = Z_{i-1}, \sigma)$
  - δ and σ are called tuning parameters and control the size of the 'jump'.

Metropolis Algorithm

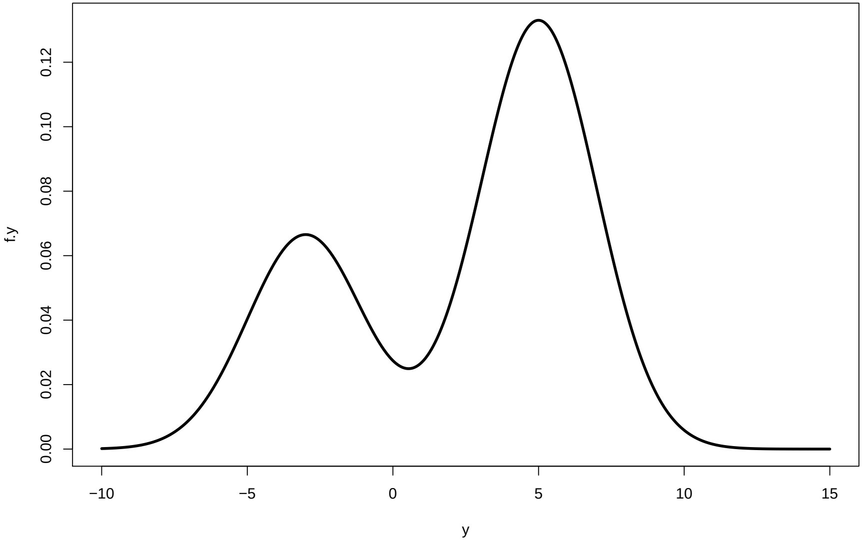
$P(\theta|y) \propto P(y|\theta)P(\theta)$

- Let's use the Metropolis algorithm to generate values from the following mixture distribution:

$$f_Y(y) = \frac{1}{3}\mathrm{normal}(\mu = -3, \sigma = 2) + \frac{2}{3}\mathrm{normal}(\mu = 5, \sigma = 2)$$

```
y <- seq(-10, 15, by=0.01)
f.y <- (1/3)*dnorm(y,-3, 2) + (2/3)*dnorm(y, 5, 2)
plot(y, f.y, type="l", lwd=3)
```

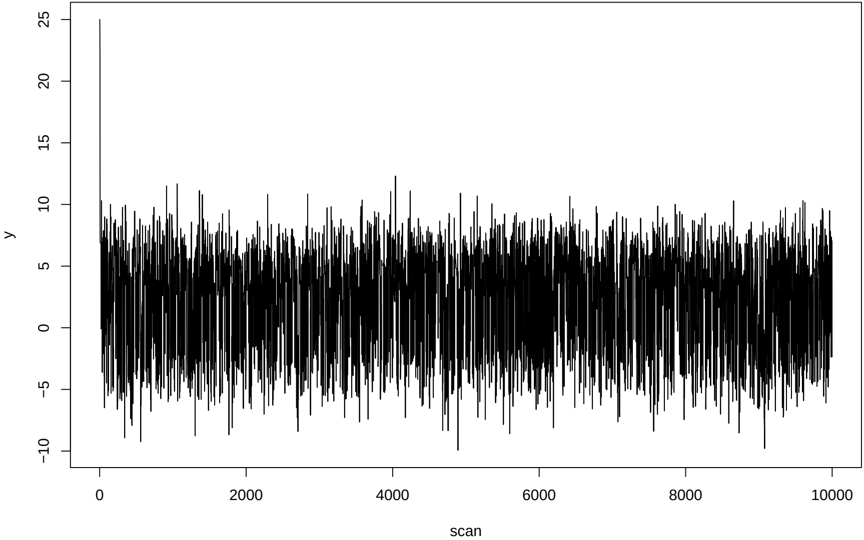
Metropolis Algorithm



Metropolis Algorithm $\delta = 10$

```
set.seed(1001)
S <- 10000
out <- rep(0, S)
acc <- 0

## density
f.y <- function(y){
  out <- (1/3)*dnorm(y,-3, 2) + (2/3)*dnorm(y, 5, 2)
  return(out)
  }

## starting value
y <- 25
out[1] <- y

## tuning parameter
delta <- 10

## MCMC
for(i in 2:S){

  y.star <- runif(1, y-delta, y+delta)
  r <-  f.y(y.star)/f.y(y)
  rho <- min(r,1)

  if(runif(1) <= rho){
    y <- y.star
    acc <- acc + 1
    }

  out[i] <- y

  }
```

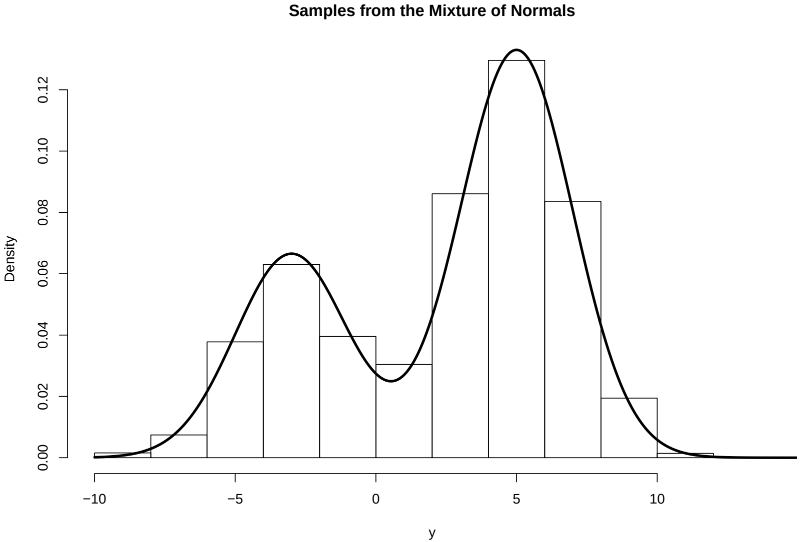
```r
plot(out, type="l", ylab="y", xlab="scan")
```



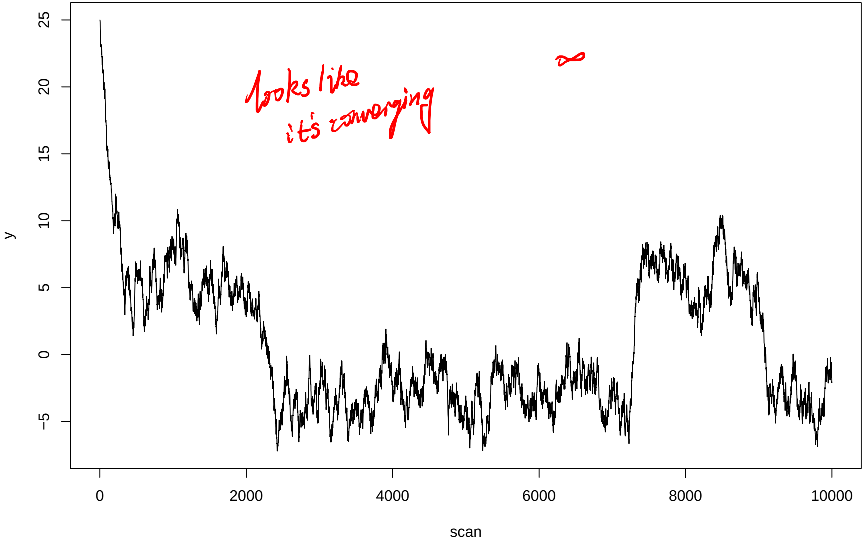
The acceptance rate was 0.5099.

- let's remove the first 100 values for burn-in.

```
hist(out[-c(1:100)], prob=TRUE,
     main="Samples from the Mixture of Normals", xlab="y")
y <- seq(-10, 15, by=0.01)
f.y <- (1/3)*dnorm(y,-3, 2) + (2/3)*dnorm(y, 5, 2)
lines(y, f.y, type="l", lwd=3)
```

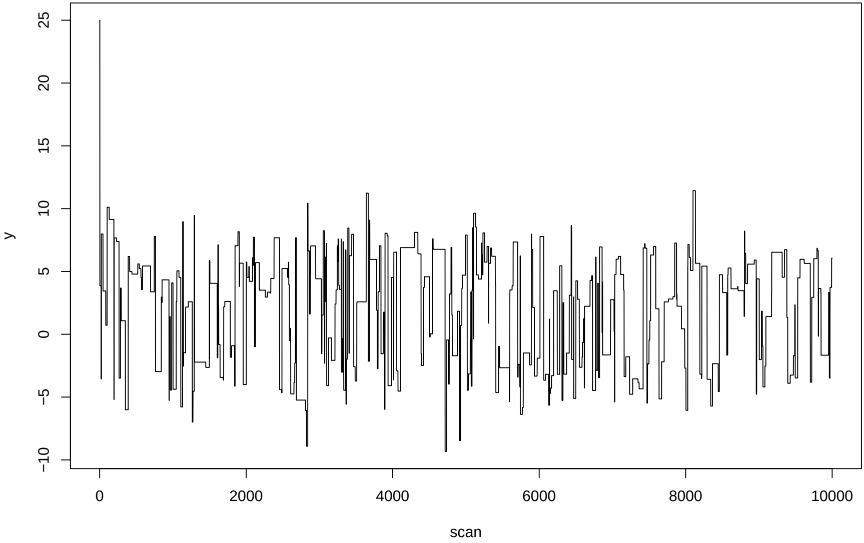

Samples from the Mixture of Normals
Density
0.00
0.02
0.04
0.06
0.08
0.10
0.12
−10
−5
0
5
10
y

Metropolis Algorithm - Small $\delta = 0.5$



The acceptance rate was 0.9566.

Metropolis Algorithm - Large $\delta = 150$



The acceptance rate was 0.0372.

MCMC

- As you might expect there are numerous variations on the Metropolis-Hastings approach in order to efficiently sample for the target distribution. See Givens and Hoeting for more information.

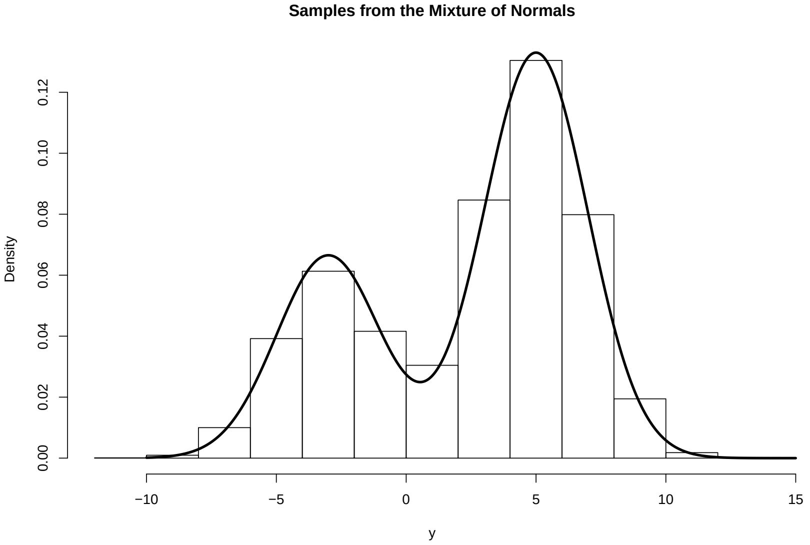
Generating Random Samples

- Based on what we know we can consider a direct approach to the simulation of the mixture of normals:
    1. Generate $X \sim \text{Bernoulli}(p = 1/3)$.
    2. If $X = 1$ generate $Z \sim \text{normal}(\mu = -3, \sigma = 2)$. If $X = 0$ generate $Z \sim \text{normal}(\mu = 5, \sigma = 2)$.

Generating Random Samples

```r
set.seed(1001)
n <- 10000
out <- rep(0, n)
x <- rbinom(n, 1, 1/3)

out[x==1] <- rnorm(length(out[x==1]), -3, 2)
out[x==0] <- rnorm(length(out[x==0]), 5, 2)
```

```r
hist(out, prob=TRUE,
     main="Samples from the Mixture of Normals", xlab="y")
y <- seq(-10, 15, by=0.01)
f.y <- (1/3)*dnorm(y,-3, 2) + (2/3)*dnorm(y, 5, 2)
lines(y, f.y, type="l", lwd=3)
```


Samples from the Mixture of Normals
Density
0.00
0.02
0.04
0.06
0.08
0.10
0.12
−10
−5
0
5
10
15
y

```r
plot(out, type="l", ylab="y", xlab="scan")
```

√x