# Statistical Inference

**Lecture 02b**

ANU - RSFAS

Last Updated: Tue Mar 7 11:42:27 2017

# Monte Carlo Integration

- Thus far we observe random variables $X_1, \ldots, X_N \sim f(x|\theta)$ and are concerned with using properties of $f(x|\theta)$ to describe the behavior of the random variables.
- Now we will generate random samples of $X$ to learn about their behavior, as well as $h(X)$.
- Monte Carlo integration:
    - Many quantities of statistical analyses can be expressed as the expectation of a function of a random variable $E[h(X)]$.
    - Let $f(X|\theta)$ denote the density of $X$
    - Let $\mu$ denote the expectation of $h(X)$.
    - Then when an iid sample $X_1, \ldots, X_n$ is obtained from $f(X|\theta)$, we can approximate $\mu$ by a sample average:

$$\hat{\mu}_{MC} = \frac{1}{n}\sum_{i=1}^{n} h(X_i) \rightarrow \int h(x)f(x)dx = \mu$$

# Monte Carlo Integration

- We can approximate $\sigma^2$ similarly:

$$\hat{\sigma}^2_{MC} = \frac{1}{n-1}\sum_{i=1}^{n}\left(h(X_i) - \hat{\mu}_{MC}\right)^2 \to \sigma^2$$

- **These results are based on the Law of Large Numbers.**

# Monte Carlo Integration

Example (Exponential lifetime):

- Suppose that a particular electrical component can be modeled with an exponential ($\beta = 50$) lifetime.

$$f(x|\beta) = \frac{1}{\beta} exp(-x/\beta)$$

- The manufacturer is interested in determining the probability that, out of $c = 100$ components, at least $t = 35$ of them will last $h = 45$ hours.

- We can first consider the analytical solution. The probability that a single component last at least $h = 45$ is:

$$p_1 = \int_{45}^{\infty} \frac{1}{50} exp(-x/50) dx = 1/exp(45/50) \approx 0.4066$$

```
set.seed(1001)
n <- 20000
x <- rexp(n, 1/50)
x[1:5]
```

```
## [1]  14.30061  34.86863 118.94469  42.38883  26.53421
```

```
mean(x)
```

```
## [1] 50.22228
```

```r
p1 <- length(x[x>=45])/n
p1
```

```
## [1] 0.4082
```

```r
mean(x>=45)
```

```
## [1] 0.4082
```

$$
\begin{aligned}
p_2 &= P(\text{at least } t = 35 \text{ components last at least } h = 45 \text{ hours}) \\
&= \sum_{t=35}^{100} \binom{100}{t} p_1^t (1 - p_1)^{100-t}
\end{aligned}
$$

```r
1-pbinom(34, 100, 0.4066)
```

```
## [1] 0.895889
```

How about at least 90 out of 100 last at least 45 hours?

```r
1-pbinom(89, 100, 0.4066)
```

```
## [1] 0
```

# Full Monte Carlo Solution

- For $j = 1, \ldots, n$:
  1. Generate $X_1, \ldots, X_{c=100} \overset{\text{iid}}{\sim} \text{exponential}(\beta = 50)$.
  2. Set $Y_j = 1$ if at least $t = 35$ $X_i$s are $\geq h = 45$; otherwise set $Y_j = 0$.

Then, because $Y_j \sim \text{Bernoulli}(p_2)$ and $E[Y_j] = p_2$,

$$\frac{1}{n} \sum_{j=1}^{n} Y_j \to p_2 \text{ as } n \to \infty$$

```r
set.seed(1001)
n <- 10000

y <- rep(0, n) ## storage
for(i in 1:n){
  x <-rexp(100, 1/50)
  if(length(x[x>=45])>=35){
  y[i] <- 1}
  }

mean(y)
```

```
## [1] 0.8949
```

We can see that being able to generate random values from various probability distributions can be quite useful!

# Generating Random Samples

- There are a number of approaches to the generation of random variables.
- Let's start by considering the simplest approach, the probability integral transform.
- For this approach (and actually most every approach I can think of) we assume that we are able to generate:

$$U_1, \ldots U_m \overset{\text{iid}}{\sim} \text{uniform}(0, 1)$$

## Rice Section 2.3 Proposition C & D (Probability Integral Transform):

- Let $X$ have a continuous cdf $F_X(x)$.
- Define the random variable $Y = F_X(x)$.
- Then $Y$ is uniformly distributed on (0,1). $P(Y \leq y) = y \quad 0 < y < 1$.

**Proof:**

$$\begin{aligned} P(Y \leq y) &= P(F_X(x) \leq y) \\ &= P(F_X^{-1}[F_X(x)] \leq F_X^{-1}[y]) \\ &= P(X \leq F_X^{-1}[y]) \\ &= F_X(F_X^{-1}[y]) = y \end{aligned}$$

Note: If $F_X$ is flat in a region then it may be that $F_X^{-1}[F_X(x)] \neq x$

- Let $x \in [x_1, x_2] \Rightarrow F_X^{-1}[F_X(x)] = x_1$ for any $x$ in the interval.
- However, $P(X \leq x) = P(X \leq x_1)$.
- Generally we just define $F_X^{-1}(y) = \inf\{x | F(x) \geq y\}$

- Simply: $X = F_X^{-1}(U)$ has the distribution $F_X$.
- Consider $X \sim \mathrm{exponential}(\beta = 2)$:

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

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

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

```r
set.seed(1001)
u <- runif(10000, 0, 1)
x <- - 2*log(1-u)

mean(x)
```

```
## [1] 1.989107
```

```r
var(x)
```

```
## [1] 3.915259
```

- $E[X] = \beta = 2, \; V(X) = \beta^2 = 4$

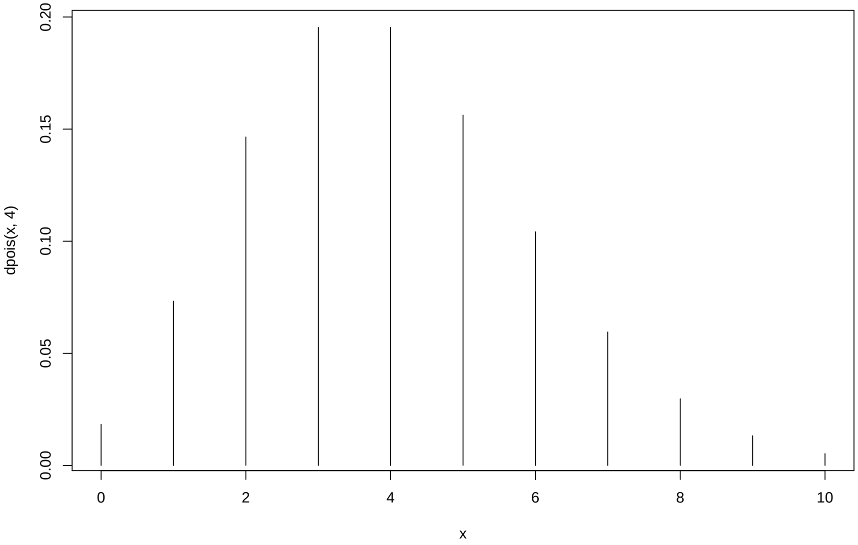
# Distributions in R

- Consider $X \sim \mathrm{Poisson}(\lambda = 4)$ (density):
- $P(X = 2)$ use 'd':

```r
dpois(2, 4)
```

```
## [1] 0.1465251
```

```r
x <- 0:10
plot(x, dpois(x, 4), type="h")
```



# Distributions in R

- $P(X \leq 2)$ use 'p' (probability):

```r
ppois(2, 4)
```

```
## [1] 0.2381033
```

- $P(X \leq x^*) = 0.25$, to find $x^*$ use 'q' (quantile):

```r
qpois(0.25, 4)
```

```
## [1] 3
```

# Distributions in R

- Remember the quantile must achieve the specified probability:

```r
ppois(2, 4)
```

```
## [1] 0.2381033
```

```r
ppois(3, 4)
```

```
## [1] 0.4334701
```

- So $x^* = 3$

# Distributions in R

- To generate random values use 'r'. Let's generate $n = 10$ random values:

```r
rpois(10, 4)
```

```
##  [1] 2 6 7 3 1 7 1 5 4 4
```

## Generating Random Samples

- For an exponential $(\beta)$ 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:

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

  - If $\beta = 2$, then the distribution is a $\chi^2$ 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:

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

## 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)
```

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

```r
var(y)
```

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

# Generating Random Samples

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

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

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

- If we could generate a normal$(0, 1)$ then we could generate a $\chi^2_1$.
- 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 $\chi^2_1$ 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 $F_Y(y) < U \leq F_Y(y_{i+1}), setY = y_{i+1}$.

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

# Generating Random Samples

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

1. Generate $U \sim \text{uniform}(0, 1)$.
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

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

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

mean(y)
```

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

```r
var(y)
```

```
## [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

$$M = \sup_y \frac{f_Y(y)}{f_V(y)} < \infty$$

- 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

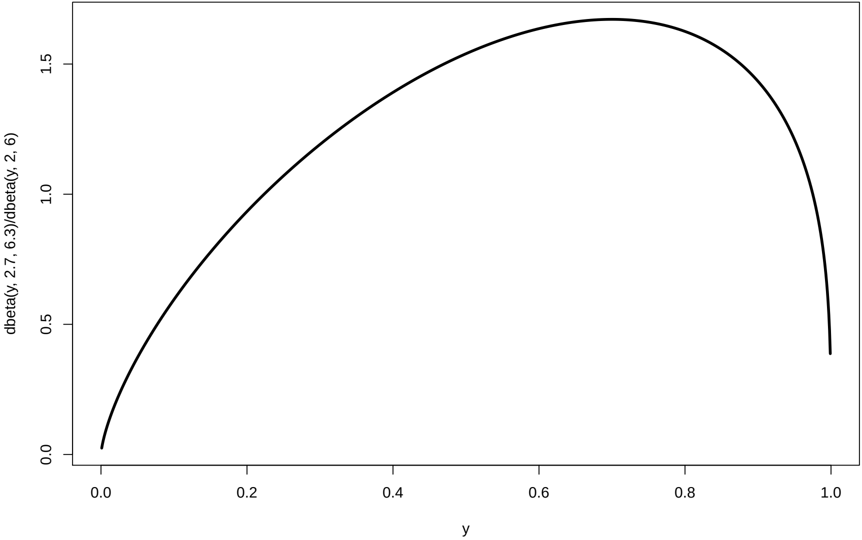
- Example 5.6.9:
  - We know how to generate $V \sim \mathrm{beta}(2, 6)$, see slide 3.
  - Now let's generate $Y \sim \mathrm{beta}(2.7, 6.3)$. The previous method will not work!
  - Lets first figure out $M$:

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

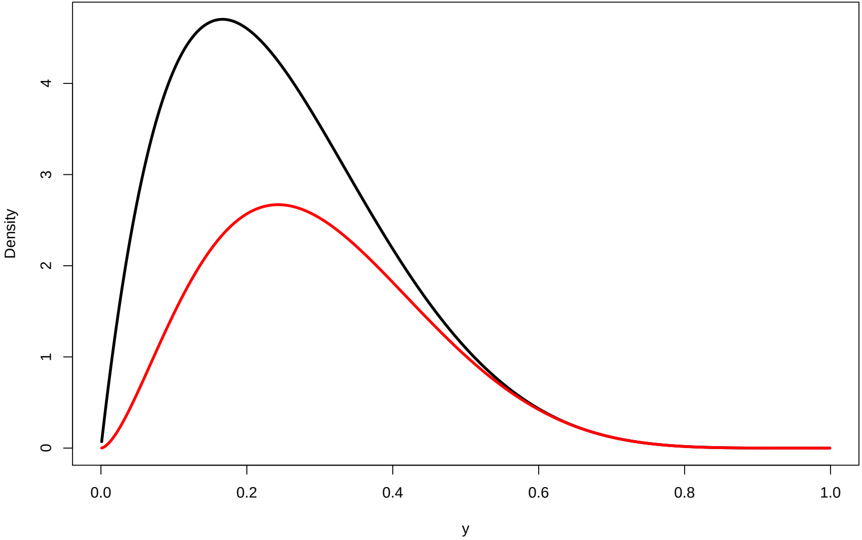
```
y <- seq(0.001, 0.999, by=0.001)
M <- max(dbeta(y, 2.7, 6.3)/dbeta(y, 2, 6))
M
```

```
## [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

```r
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)
```

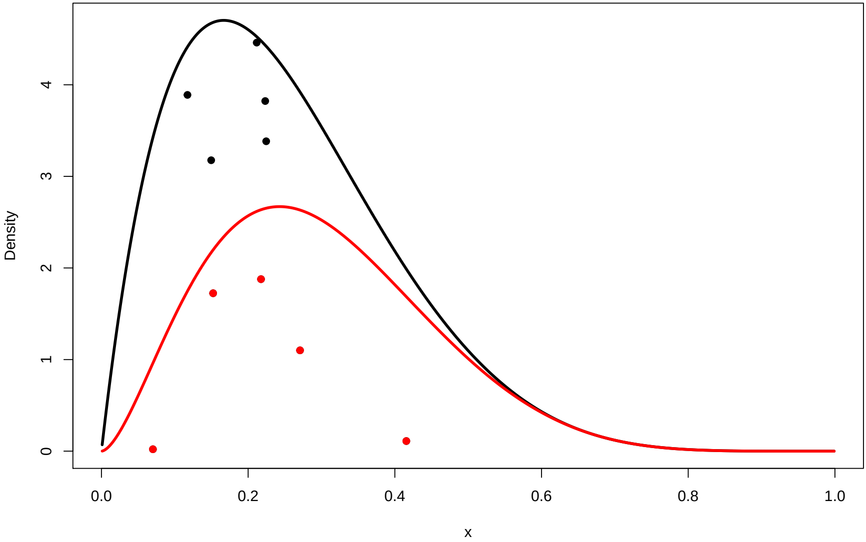
```
## [1] 6039
```

# First 10 Draws

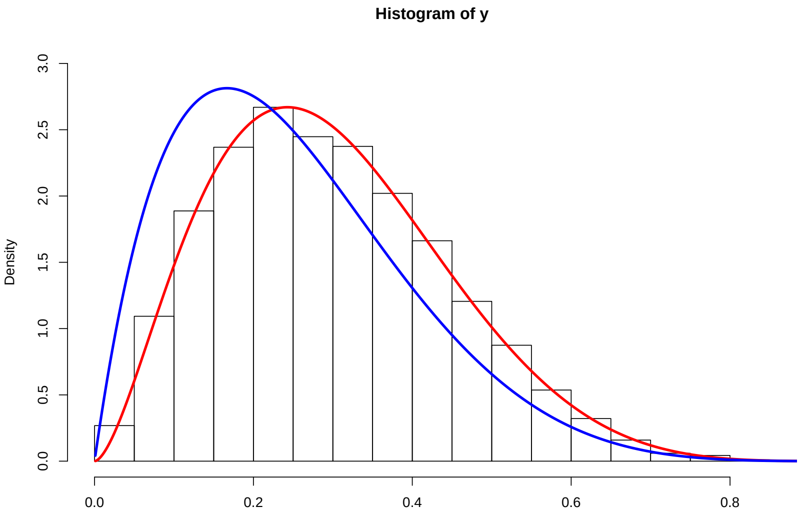
```
set.seed(1001)
n <- 10

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



```r
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")
```



# Generating Random Samples

**Proof:**

$$
\begin{aligned}
P(Y \leq y) &= P\left(V \leq y|\mathrm{stop}\right) \\
&= P\left(V \leq y|U < \frac{1}{M}\frac{f_Y(V)}{f_V(V)}\right) \\
&= \frac{P\left(V \leq y \text{ and } U < \frac{1}{M}\frac{f_Y(V)}{f_V(V)}\right)}{P\left(U < \frac{1}{M}\frac{f_Y(V)}{f_V(V)}\right)} \\
&= \frac{\int_{-\infty}^{y}\int_0^{\frac{1}{M}\frac{f_Y(v)}{f_V(v)}} f_U(u)f_V(v)dudv}{\int_{-\infty}^{\infty}\int_0^{\frac{1}{M}\frac{f_Y(v)}{f_V(v)}} f_U(u)f_V(v)dudv} = \frac{\int_{-\infty}^{y}\int_0^{\frac{1}{M}\frac{f_Y(v)}{f_V(v)}} 1 f_V(v)dudv}{\int_{-\infty}^{\infty}\int_0^{\frac{1}{M}\frac{f_Y(v)}{f_V(v)}} 1 f_V(v)dudv} \\
&= \frac{\int_{-\infty}^{y}\frac{1}{M}f_Y(v)dv}{\frac{1}{M}} = \int_{-\infty}^{y} f_Y(v)dv
\end{aligned}
$$

# Generating Random Samples

- What can we say about $M$?

$$
\begin{aligned}
P\,(\mathrm{stop}) &= P\left(U < \frac{1}{M}\frac{f_Y(V)}{f_V(V)}\right) \\
&= \int_{-\infty}^{\infty}\int_{0}^{\frac{1}{M}\frac{f_Y(v)}{f_V(v)}} f_U(u) f_V(v)\ du\ dv = \int_{-\infty}^{\infty}\int_{0}^{\frac{1}{M}\frac{f_Y(v)}{f_V(v)}} 1\ du\ f_V(v) dv \\
&= \int_{-\infty}^{\infty} \frac{1}{M}\frac{f_Y(v)}{f_V(v)} f_V(v) dv \\
&= \int_{-\infty}^{\infty} \frac{1}{M} f_Y(v) dv \\
&= \frac{1}{M}\int_{-\infty}^{\infty} f_Y(v) dv \\
&= \frac{1}{M} \times 1 = \frac{1}{M}
\end{aligned}
$$

# Generating Random Samples

- We are considering the number of trials till a success (a geometric distribution). If $W \sim \mathrm{geometric}(\theta)$ 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

- 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 distirbution 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

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

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:
  - $f_V(Y_i^*|Z_{i-1}) = \mathrm{uniform}(Z_{i-1} - \delta, Z_{i-1} + \delta)$
  - $f_V(Y_i^*|Z_{i-1}) = \mathrm{normal}(\mu = Z_{i-1}, \sigma)$
  - $\delta$ and $\sigma$ are called tuning parameters and control the size of the 'jump'.

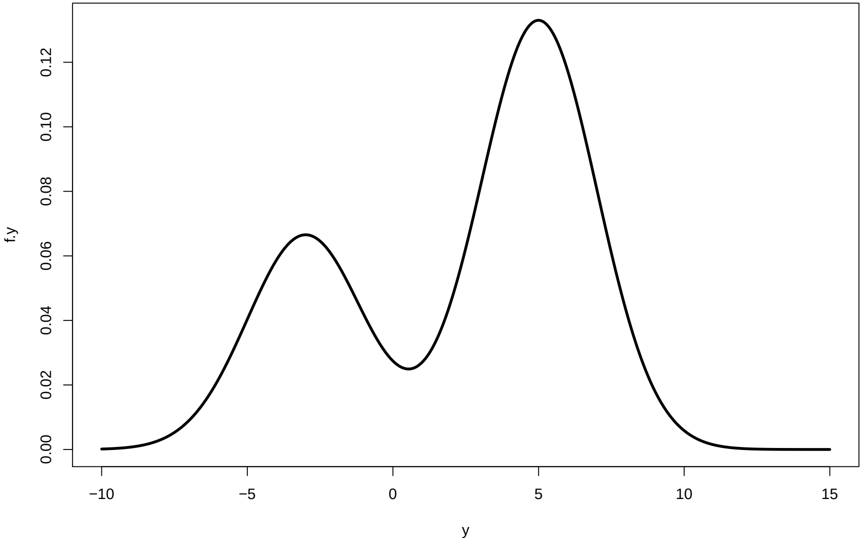
# Metropolis Algorithm

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

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

```r
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)
```

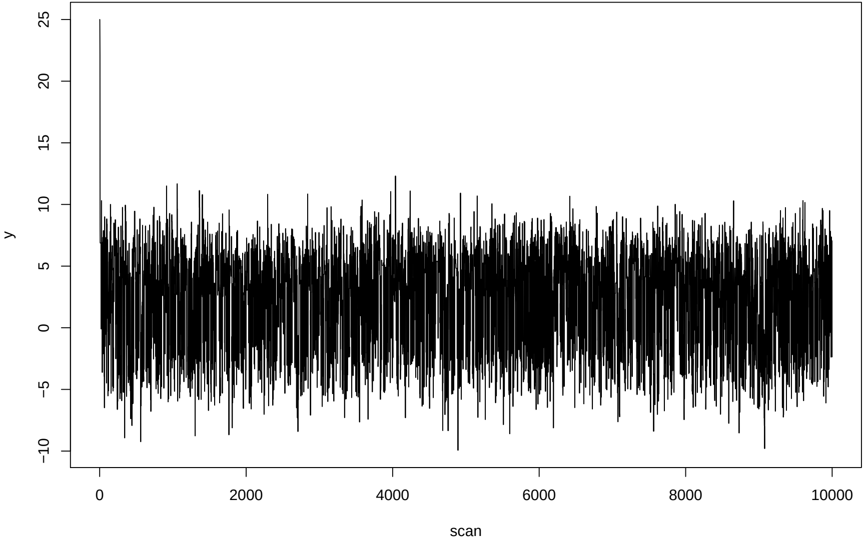
# Metropolis Algorithm



# 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

  }
```

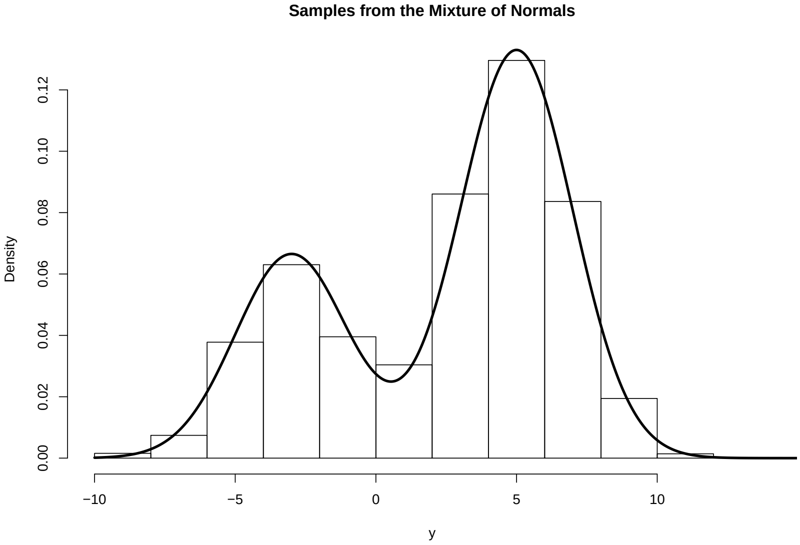
```r
plot(out, type="l", ylab="y", xlab="scan")
```



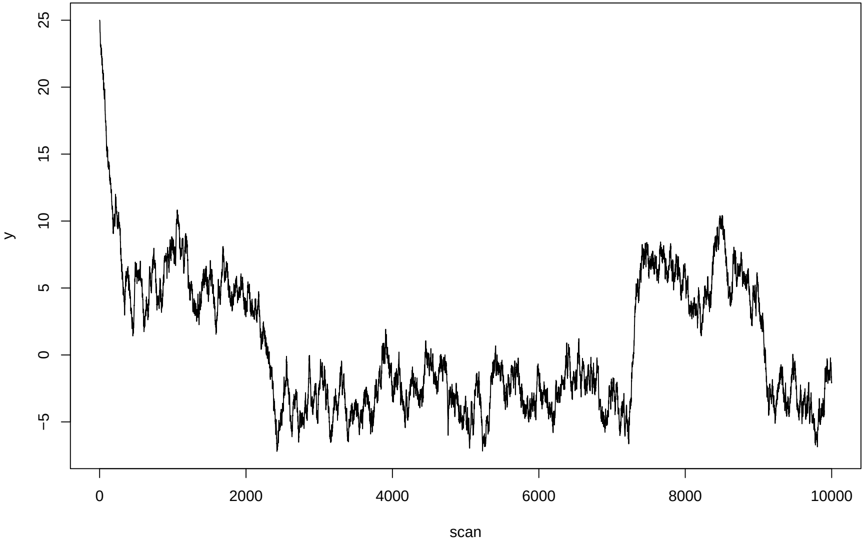

The acceptance rate was 0.5099.

- 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)
```

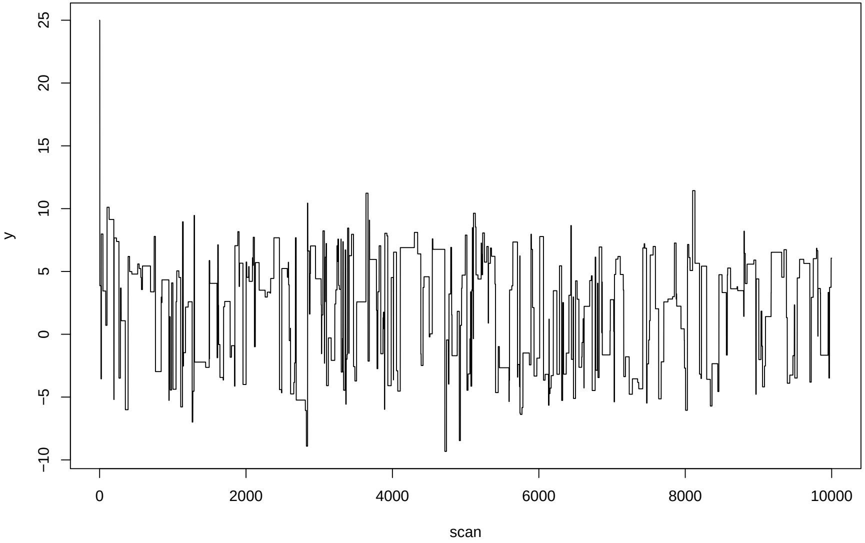

Samples from the Mixture of Normals
Density
0.00
0.02
0.04
0.06
0.08
0.10
0.12
−10
−5
0
5
10
y

# 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.

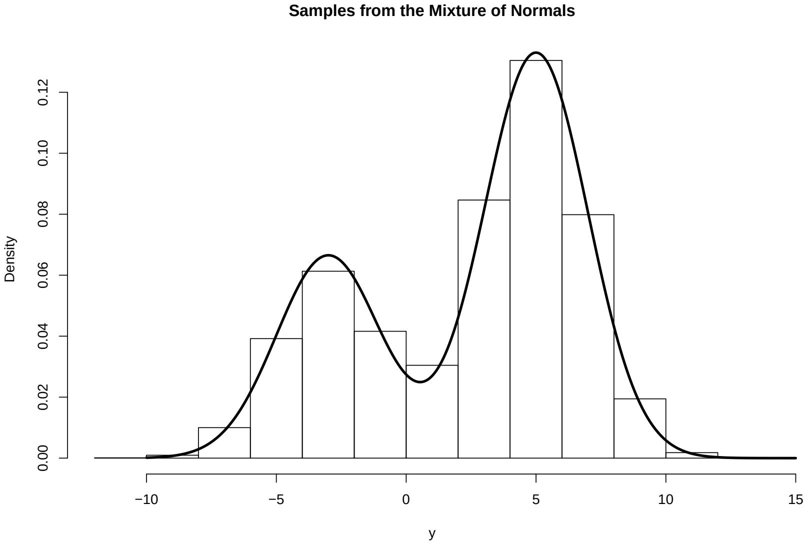
# Generating Random Samples

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

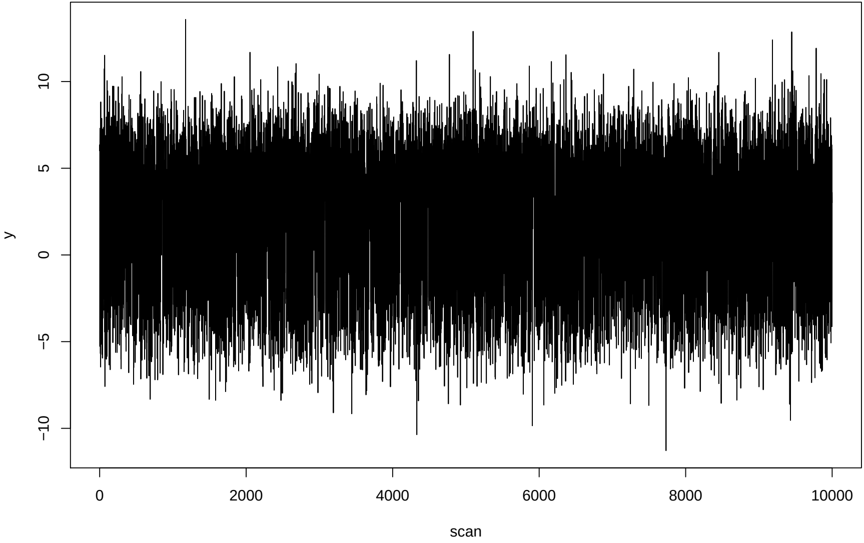
# Generating Random Samples

```
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)
```

```
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")
```



## Moving Forward

- In Chapter 6 we will cover:
  - Distirbutions derived from normal distributions