Metropolis Hastings

Jonathan Navarrete

General Metropolis-Hastings

For a given target density *f* we wish to estimate, we build a Markov kernel *K* with stationary distribution *f* and then generate a Markov chain X_t using this kernel so that the limiting distribution of the Markov chain is *f* and integrals can be approximated according to the *Ergodic Theorem*.

The **Metropolis-Hastings algorithm** is a general purpose MCMC method for approximating a target density *f*, using a conditional density $q(y|x)$ that is easy to simulate from.

In addition, q can be almost arbitrary in that the only theoretical requirements are that the ratio $\frac{f(y)}{q(y|x)}$ is known up to a constant *independent* of x and that $q(\cdot|x)$ has enough dispertion to lead to an exploration of the entire support of *f*

We can rely on the feature of Metropolis-Hatings algorithm that for every given *q*, we can then construct a Metropolis-Hastings kernel such that *f* is its stationary distribution.

General Metropolis-Hastings

The Metropolis-Hastings algorithm as described Robert & Casella goes as follows

Given $x^{(t)}$

```
1. Generate $Y_t \sim q(y|x_t)$
2. Take
```

$$X_{t+1} = \begin{cases} Y_t & \text{with probability } \rho(x^{(t)}, Y_t) \\ x^{(t)} & \text{with probability } 1 - \rho(x^{(t)}, Y_t) \end{cases}$$

where

$$\rho(x^{(t)}, Y_t) = \min\left\{ \frac{f(Y_t)}{f(x^{(t)})} \frac{q(x^{(t)}|Y_t)}{q(Y_t|x^{(t)})}, 1 \right\}$$

General Metropolis-Hastings

In simpler terms, as we want to generate $X \sim f$, we first take an initial value $x^{(0)}$ (which can almost be any artibrary value in the support of f).

1. We generate a value $Y_0 \sim q(y|x^{(0)})$.
2. We calculate $\rho(x^{(t)}, Y_t)$
3. Generate a random value $U \sim Unif(0,1)$
4. If $U < \rho(x^{(t)}, Y_t)$, then we accept $X^{(1)} = Y_t$; else we take $X^{(1)} = X^{(0)}$
5. Repeate steps 1-4 until you've satisfied the number of samples needed

General Metropolis-Hastings

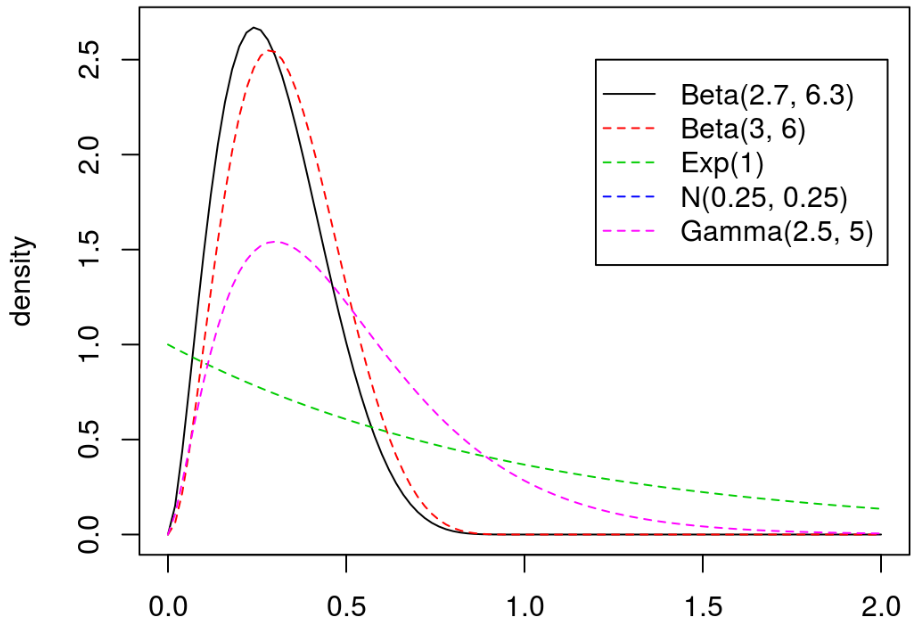
You may notice the MH algorithm is not too dissimilar from the Accept-Reject algorithm.

```
1. Generate $Y \sim g$, $U \sim Unif(0,1)$
2. Accept $X = Y$ if $U \leq \frac{f(Y)}{Mg(Y)}$;
3. Return to step 1 otherwise
```

Example Beta(2.7, 6.3)

As of now, we've covered multiple ways of generating random samples from a target density. Let us compare the accept-reject algorithm once more with the Metropolis-Hastings algorithm. Generate N samples from $Beta(2.7, 6.3)$

In order to use the accept-reject algorithm, we need a candidate distribution to sample from. Below are a set of potential candidate distributions.



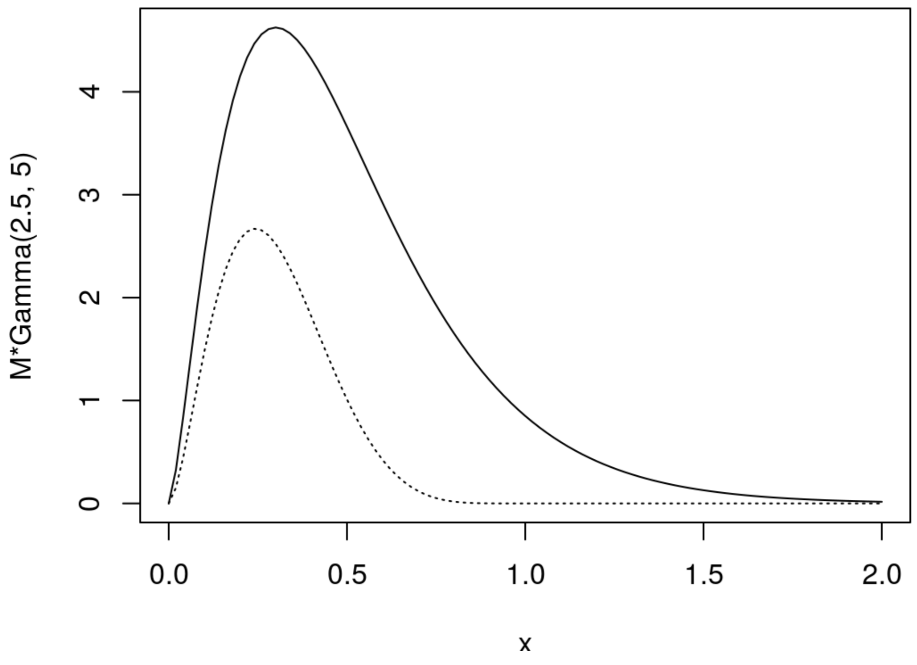
6/50

Example Beta(2.7, 6.3)

```
## potential instumential distributions
curve(expr = dbeta(x, 2.7, 6.3), from = 0, to = 2)
curve(expr = dbeta(x, 3, 6), from = 0, to = 2, add = TRUE, col = 2, lty = 2)
curve(expr = dexp(x, rate = 3), from = 0, to = 2, add = TRUE, col = 3, lty = 2)
curve(expr = dnorm(x, mean = 0.25, 0.25), from = -2, to = 2, add = TRUE, col = 4, lty = 2)
curve(expr = dgamma(x, shape = 2.5, scale = 1/5), from = 0, to = 2, add = TRUE, col = 6, lty = 2)
legend(x = 1.2, y = 2.5, legend = c("Beta(2.7, 6.3)", "Beta(3, 6)", "Exp(1)", "N(0.25, 0.25)", "Gamma(2.5, 5)
       lty = c(1, 2, 2, 2, 2), col = c(1:4, 6), merge = TRUE)
```

Example Beta(2.7, 6.3)

```
M = 3
par(pin = c(4.5, 3))
curve(expr = M*dgamma(x, shape = 2.5, scale = 1/5), from = 0, to = 2, ylab = "M*Gamma(2.5, 5)")
curve(expr = dbeta(x, 2.7, 6.3), from = 0, to = 2, add = TRUE, lty = 3)
```



We will use $Exp(3)$ as our candidate distribution, g.

Example Beta(2.7, 6.3)

```
set.seed(1234)
N = 500000
## For accept-reject, we need to find a value for M

f = function(x){
  dbeta(x, 2.7, 6.3)
}

g = function(x){
  dgamma(x, shape = 2.5, scale = 1/5)
}
```

Example Beta(2.7, 6.3)

```
X = numeric(N)
i = 0
while(i < N){
  Y = rgamma(n = 1, shape = 2.5, scale = 1/5)
  U = runif(n = 1)
  if(U*M <= f(Y)/g(Y)){
    i = i + 1
    X[i] = Y
  }
}

qbeta(p = c(0, 0.25, 0.5, 0.75, 1), shape1 = 2.7, shape2 = 6.3) ## quantiles from Beta(2.7, 6.3)
```

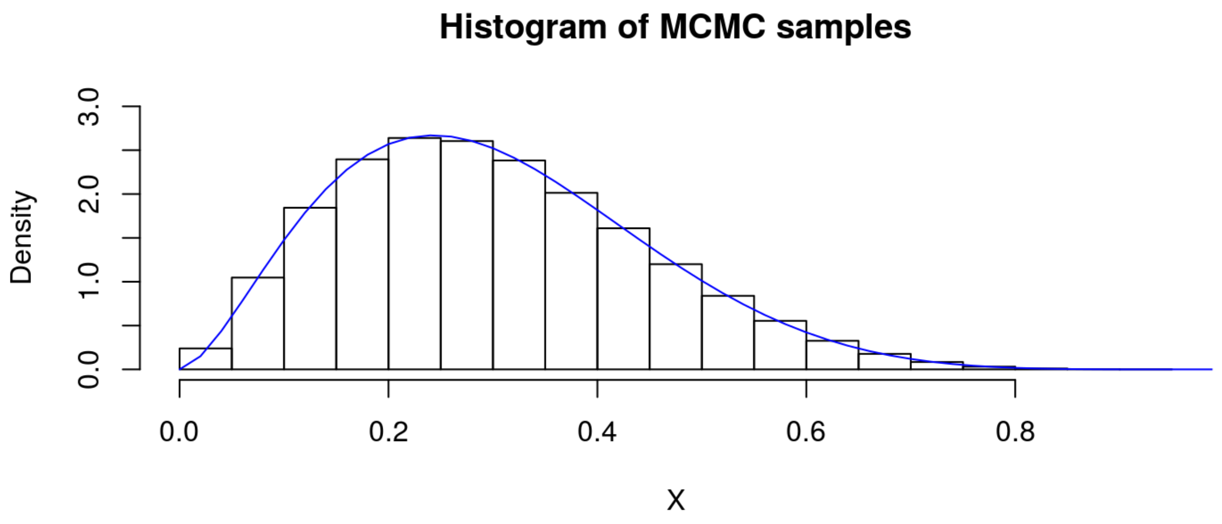
```
## [1] 0.0000000 0.1895571 0.2846608 0.3950333 1.0000000
```

```
quantile(X) ## sample mean from Accept-Reject samples
```

```
##         0%        25%        50%        75%       100%
## 0.00117328 0.18979491 0.28480053 0.39559864 0.94743365
```

Example Beta(2.7, 6.3)

```
## see how samples from chain compare to Beta(2.7, 6.3) density
hist(X, main = "Histogram of MCMC samples", prob = TRUE, ylim = c(0, 3))
curve(expr = dbeta(x, 2.7, 6.3),  from = 0, to = 2, add = TRUE, col = "blue")
```



11/50

Example Beta(2.7, 6.3)

Below is the Metropolis-Hastings implementation for this problem.

```
X = numeric(N)
X[1] = rbeta(n = 1, shape1 = 2.7, shape2 = 6.3) ## initial value
for(i in 1:N){
  Y = rgamma(n = 1, shape = 2.5, scale = X[i])  #rexp(n = 1, rate = X[i])
  rho = (dbeta(x = Y, 2.7, 6.3) * dgamma(x = X[i], shape = 2.5, scale = Y) ) /
    (dbeta(x = X[i], 2.7, 6.3) * dgamma(x = Y, shape = 2.5, scale = X[i])  )

  if(runif(1) < rho){
    X[i+1] = Y
  } else{
    X[i+1] = X[i]
  }
}
qbeta(p = c(0, 0.25, 0.5, 0.75, 1), shape1 = 2.7, shape2 = 6.3) ## quantiles from Beta(2.7, 6.3)
```

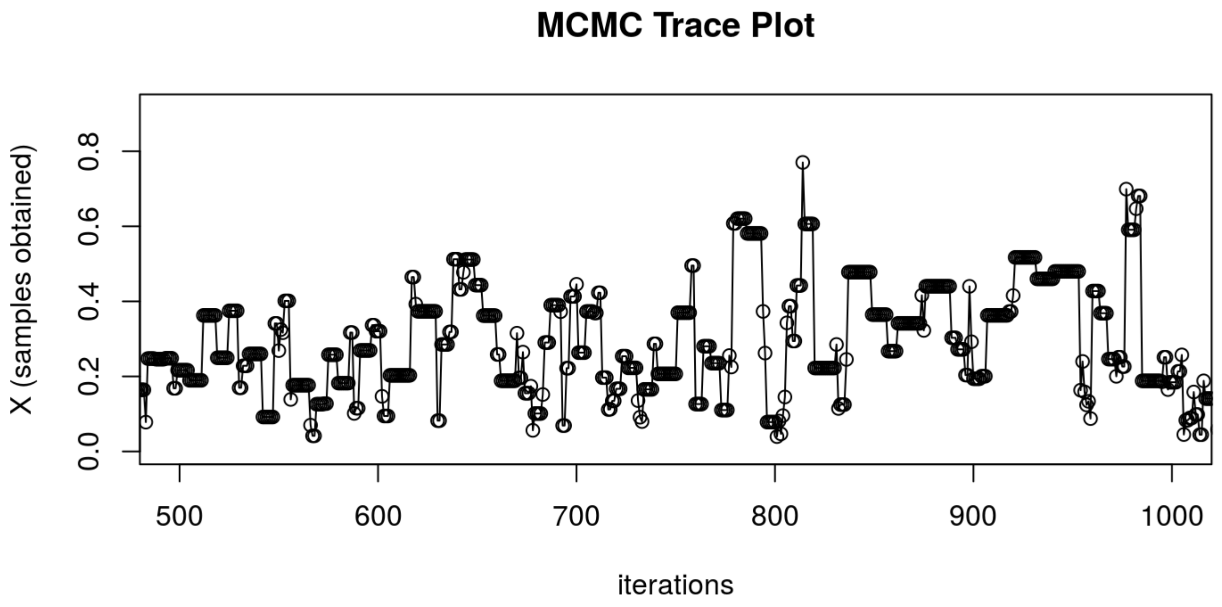
```
## [1] 0.0000000 0.1895571 0.2846608 0.3950333 1.0000000
```

```
quantile(X) ## sample mean from M-H samples
```

```
##          0%         25%         50%         75%        100%
## 0.002143683 0.189484732 0.283584815 0.392688753 0.914997756
```

Example Beta(2.7, 6.3)

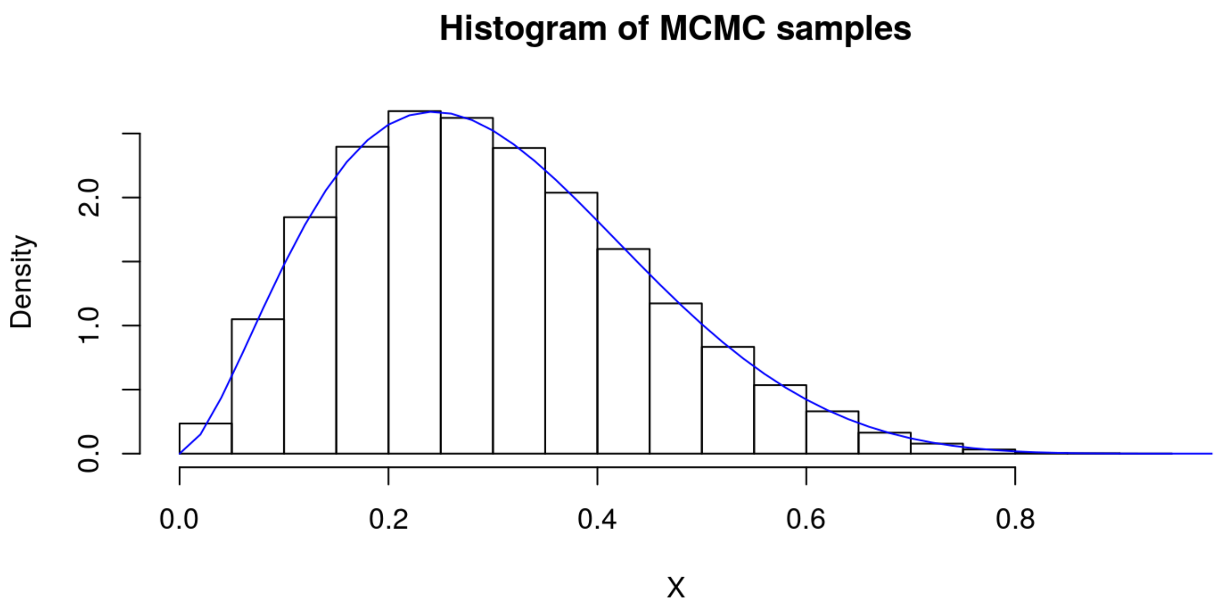
```
plot(X, type = "o", main = "MCMC Trace Plot", xlim = c(500,1000),
     xlab = "iterations", ylab = "X (samples obtained)")
```



13/50

Example Beta(2.7, 6.3)

```r
## see how samples from chain compare to Beta(2.7, 6.3) density
hist(X, main = "Histogram of MCMC samples", prob = TRUE)
curve(expr = dbeta(x, 2.7, 6.3),
      from = 0, to = 2, add = TRUE, col = "blue")
```



Example Beta(2.7, 6.3)

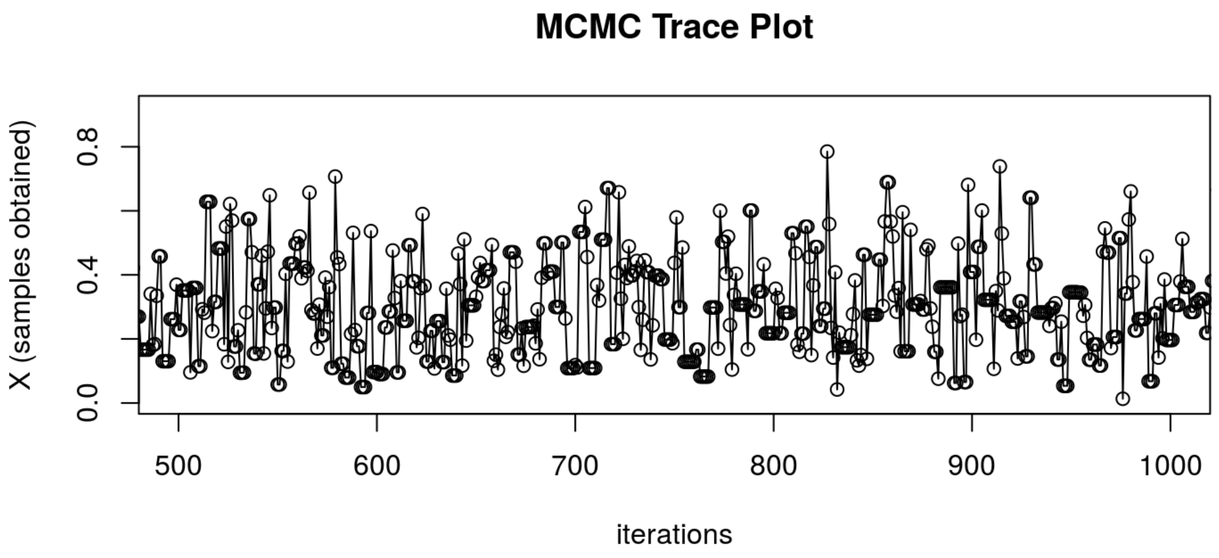
Here is a varition of the M-H algorithm used previously, except we do not let the candidate distribution depend on previous values of the chain. The candidate distribution depends only on present values of the chain, in effect $q(y|x) = q(y)$.

```
X = numeric(N)
X[1] = rbeta(n = 1, shape1 = 2.7, shape2 = 6.3)
for(i in 1:N){
  Y = rgamma(n = 1, shape = 2.5, scale = 1/5)
  rho = (dbeta(x = Y, 2.7, 6.3) * dgamma(x = X[i], shape = 2.5, scale = 1/5) ) /
    (dbeta(x = X[i], 2.7, 6.3) * dgamma(x = Y, shape = 2.5, scale = 1/5) )

  if(runif(1) < rho){
    X[i+1] = Y
  } else{
    X[i+1] = X[i]
  }

}

quantile(X)
```

```
##          0%         25%         50%         75%        100% 
## 0.002710676 0.188878363 0.284491099 0.394478021 0.922120535
```

Example Beta(2.7, 6.3)

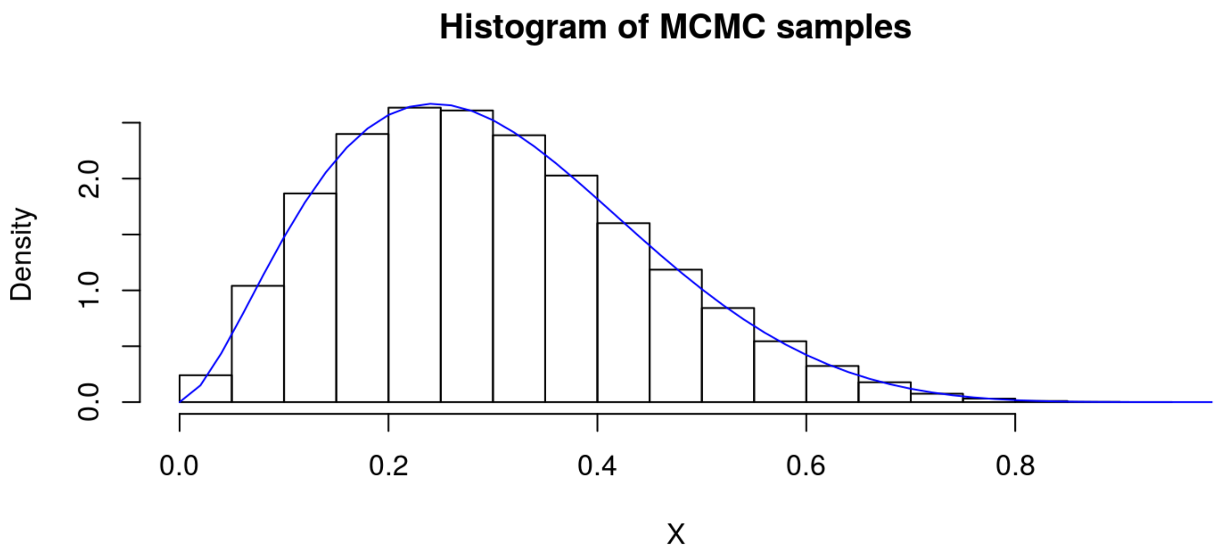
```
## see chain transitions
plot(X, type = "o", main = "MCMC Trace Plot", xlim = c(500,1000),
     xlab = "iterations", ylab = "X (samples obtained)")
```



16/50

Example Beta(2.7, 6.3)

```r
## see how samples from chain compare to Beta(2.7, 6.3) density
hist(X, main = "Histogram of MCMC samples", prob = TRUE)
curve(expr = dbeta(x, 2.7, 6.3), from = 0, to = 2, add = TRUE, col = "blue")
```



17/50

Example Beta(2.7, 6.3)

This version of the M-H algorithm is known as the **Independent Metropolis-Hastings**. This method appears a generalization of the accept-reject algorithm in the sense that the instrumental distribution is the same density g as in the accept-reject algorithm. Thus, the proposed values Y_i are the same, if not the accepted ones.

Independent M-H

The **Independent Metropolis-Hastings algorithm** as described Robert & Casella goes as follows

Given $x^{(t)}$

1. Generate $Y_t \sim g(y)$

2. Take

$$X_{t+1} = \begin{cases} Y_t & \text{with probability } \; \rho(x^{(t)}, Y_t) \\ x^{(t)} & \text{with probability } \; 1 - \rho(x^{(t)}, Y_t) \end{cases}$$

where

$$\rho(x^{(t)}, Y_t) = \min\left\{ \frac{f(Y_t)}{f(x^{(t)})} \frac{g(x^{(t)})}{g(Y_t)}, 1 \right\}$$

Independent M-H

In simpler terms, as we want to generate $X \sim f$, we first take an initial value $x^{(0)}$ (which can almost be any artibrary value in the support of f).

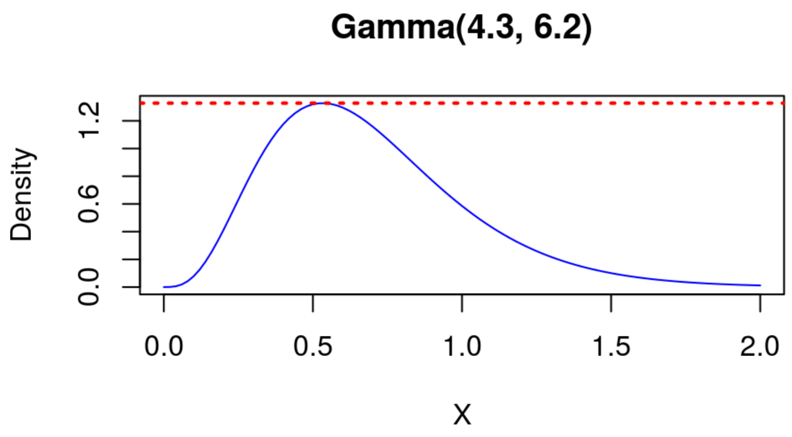
1. We generate a value $Y_0 \sim q(y|x^{(0)})$.
2. We calculate $\rho(x^{(t)}, Y_t)$
3. Generate a random value $U \sim Unif(0,1)$
4. If $U < \rho(x^{(t)}, Y_t)$, then we accept $X^{(1)} = Y_t$; else we take $X^{(1)} = X^{(0)}$
5. Repeate steps 1-4 until you've satisfied the number of samples needed

20/50

Example: Gamma(4.3, 6.2)

Here we will compare again the Accept-Reject algorithm against the Metropolis-Hastings. Generate *N* random variables $X \sim Gamma(4.3, 6.2)$.

```
## For accept-reject, we need to find a value for M
## we can use `optimize` to find the maximum of our target density
maximum  = optimize(f = function(x){ dgamma(x = x, shape = 4.3, rate = 6.2)},
                    interval = c(0, 2), maximum = TRUE ) ## obtain maximum

M = maximum$objective
curve(expr = dgamma(x = x, shape = 4.3, rate = 6.2), from = 0,
      to = 2, col = "blue", main = "Gamma(4.3, 6.2)", xlab = "X", ylab = "Density")
abline(h = M, lty = 3, lwd = 2, col = "red")
```



21/50

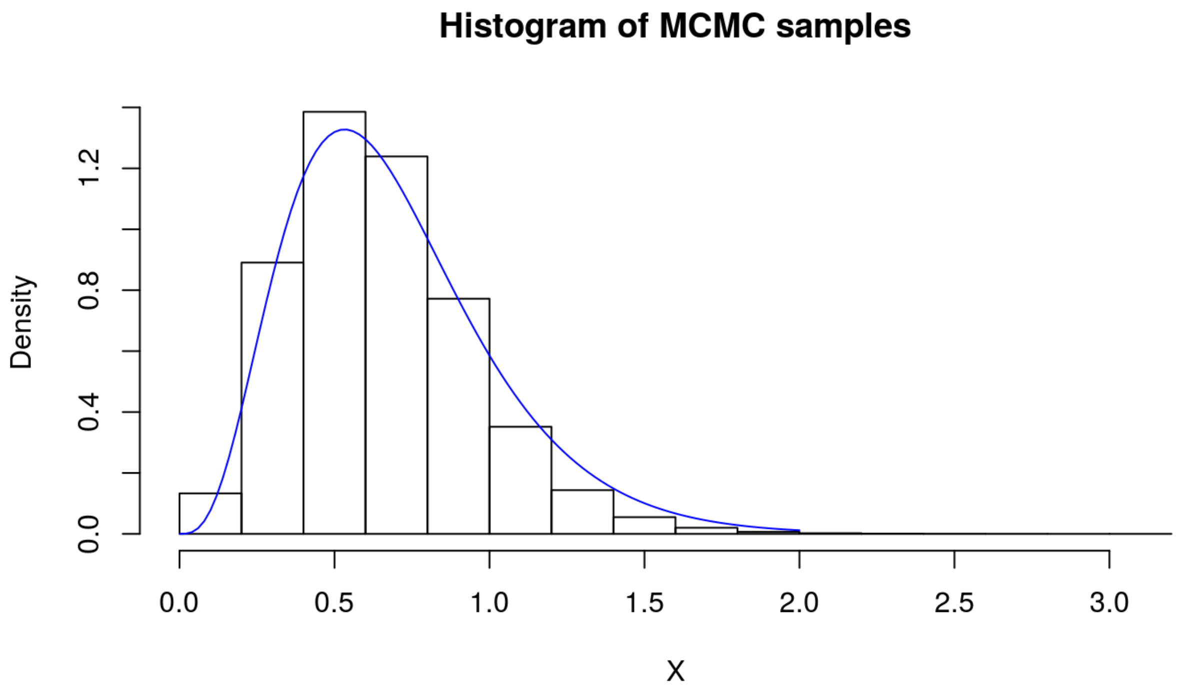
Example: Gamma(4.3, 6.2)

```
f = function(x){
  dgamma(x = x, shape = 4.3, rate = 6.2)
}

g = function(x){
  dgamma(x = x, shape = 4, rate = 7)
}

X = numeric(N)
i = 0
while(i < N){
  Y = rgamma(n = 1, shape = 4, rate = 7)
  U = runif(1)
  if(U*M <= f(Y)/g(Y)){
    i = i + 1
    X[i] = Y
  }
}
```

22/50

Example: Gamma(4.3, 6.2)

```
## see how samples from chain compare to Gamma density
hist(X, main = "Histogram of MCMC samples", prob = TRUE)
curve(expr = dgamma(x = x, shape = 4.3, rate = 6.2),
      from = 0, to = 2, add = TRUE, col = "blue")
```



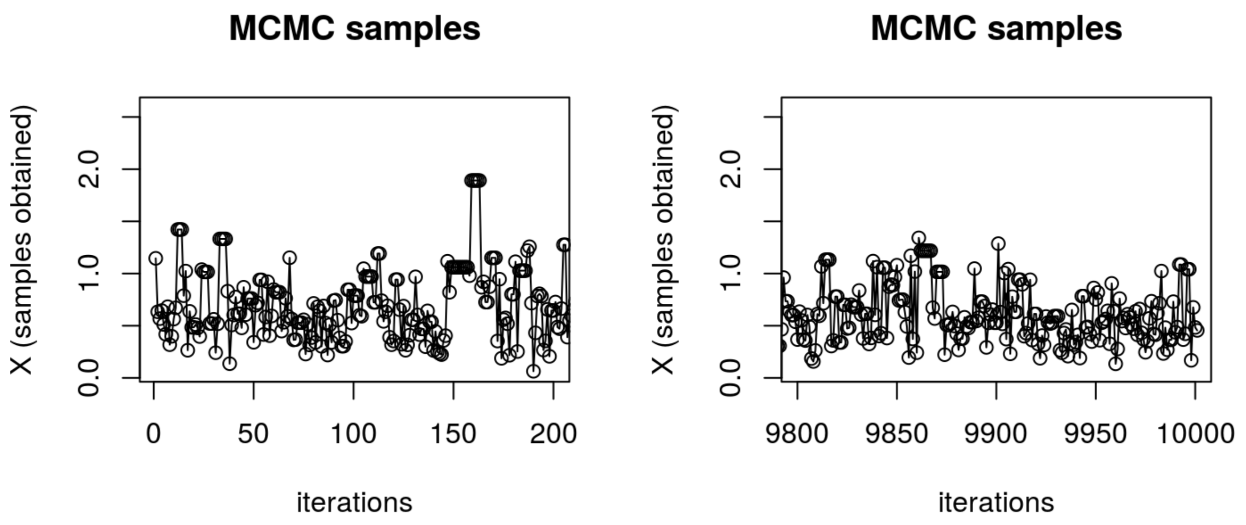
Example: Gamma(4.3, 6.2)

```r
## Metropolis Hastings

N = 10000
X = numeric(N)
X[1] = rgamma(n = 1, shape = 4.3, rate = 6.2)
for(i in 1:N){
  Y = rgamma(n = 1, shape = 4, rate = 7)
  rho = (dgamma(x = Y, shape = 4.3, rate = 6.2) * dgamma(x = X[i], shape = 4, rate = 7)) /
    (dgamma(x = X[i], shape = 4.3, rate = 6.2) * dgamma(x = Y, shape = 4, rate = 7))
  #X[i+1] = X[i] + (Y - X[i])*(runif(1) < rho) ## equivalent to if-else statement below
  if(runif(1) < rho){
    X[i+1] = Y
  } else{
    X[i+1] = X[i]
  }
}
#qgamma(p = c(0, 0.25, 0.5, 0.75, 1), shape = 4.3, rate = 6.2) #[1] 0.0000000 0.4488888 0.6405895 0.8808118
quantile(X)  ## rgamma: 0.6979356
```

```
##         0%        25%        50%        75%       100%
## 0.06364308 0.44650858 0.63902685 0.87142726 2.58634991
```

Example: Gamma(4.3, 6.2)

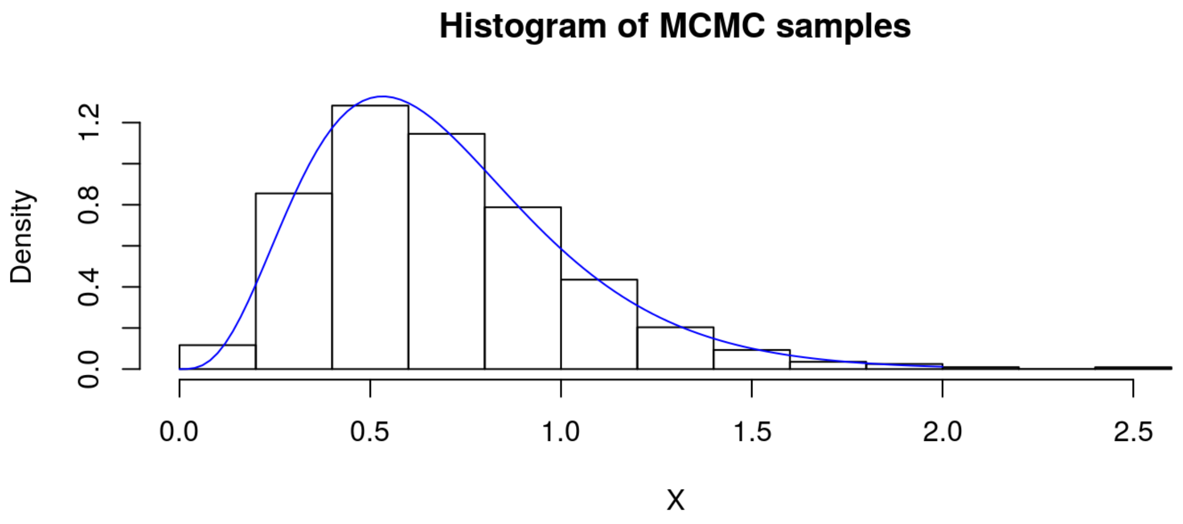
```
## see chain transitions
par(mfrow = c(1,2))
plot(X, type = "o", main = "MCMC samples",
     xlim = c(1,200),
     xlab = "iterations", ylab = "X (samples obtained)")

plot(X, type = "o", main = "MCMC samples",
     xlim = c(N-200,N),
     xlab = "iterations", ylab = "X (samples obtained)")
```



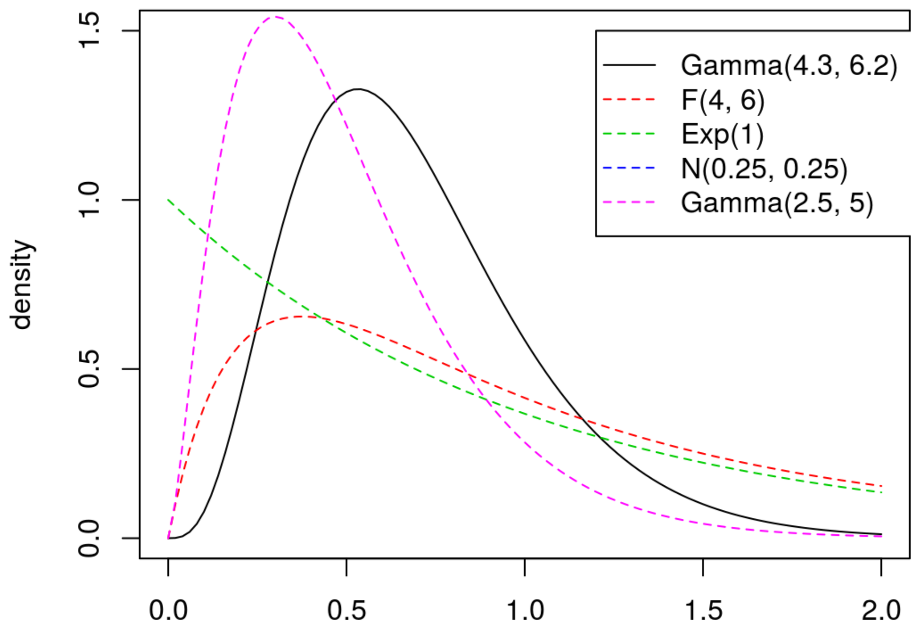
25/50

Example: Gamma(4.3, 6.2)

```r
## see how samples from chain compare to Gamma density
hist(X, main = "Histogram of MCMC samples", prob = TRUE)
curve(expr = dgamma(x = x, shape = 4.3, rate = 6.2),
      from = 0, to = 2, add = TRUE, col = "blue")
```



Example: Gamma(4.3, 6.2)



Example: Gamma(4.3, 6.2)

```
## Metropolis Hastings
X = numeric(N)
X[1] = 0.5
for(i in 1:N){
  Y = rf(n = 1, df1 = 4, df2 = 6)
  rho = (dgamma(x = Y, shape = 4.3, rate = 6.2) * df(x = X[i], df1 = 4, df2 = 6)) /
    (dgamma(x = X[i], shape = 4.3, rate = 6.2) * df(x = Y, df1 = 4, df2 = 6))
  #X[i+1] = X[i] + (Y - X[i])*(runif(1) < rho) ## equivalent to if-else statement below
  if(runif(1) < rho){
    X[i+1] = Y
  } else{
    X[i+1] = X[i]
  }
}
qgamma(p = c(0, 0.25, 0.5, 0.75, 0.9), shape = 4.3, rate = 6.2)
```

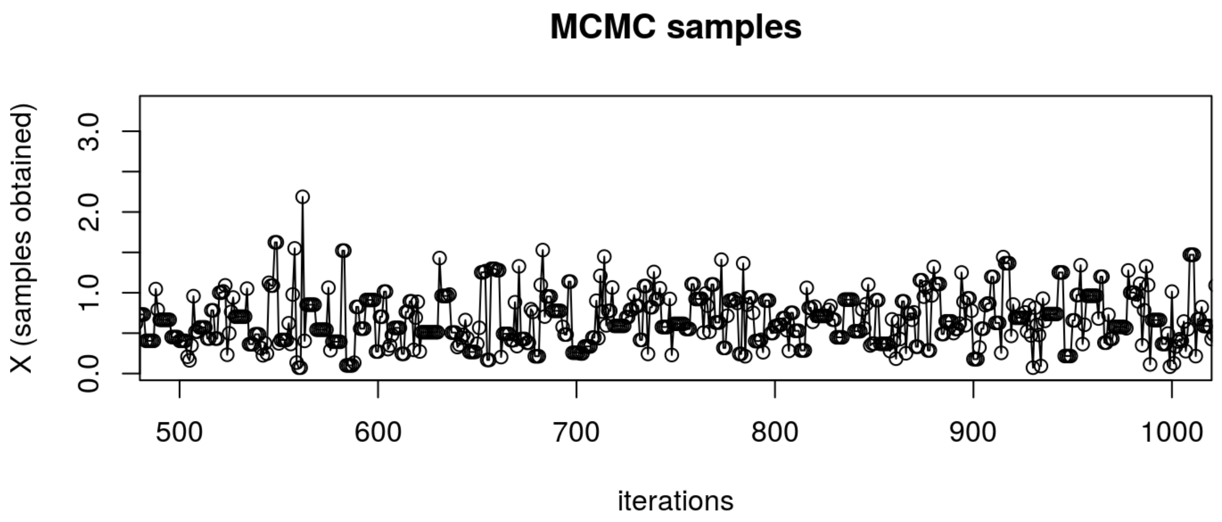
```
## [1] 0.0000000 0.4488888 0.6405895 0.8808118 1.1417128
```

```
quantile(X, probs = c(0, 0.25, 0.5, 0.75, 0.9))
```

```
##        0%       25%       50%       75%       90%
## 0.0467881 0.4479436 0.6439269 0.8912774 1.1568716
```

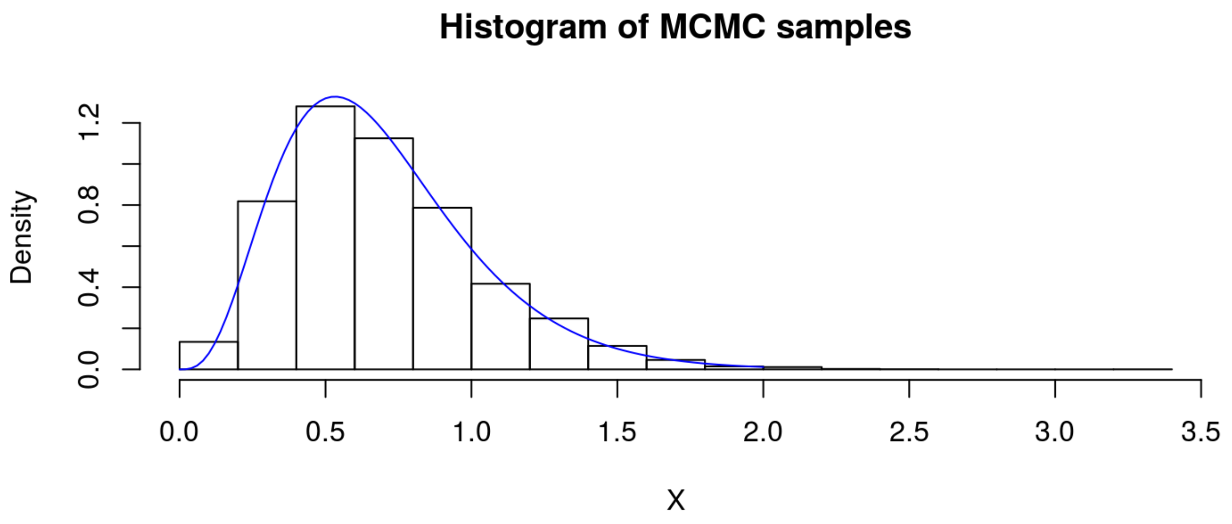
Example: Gamma(4.3, 6.2)

```
## see chain transitions
plot(X, type = "o", main = "MCMC samples", xlim = c(500,1000),
     xlab = "iterations", ylab = "X (samples obtained)")
```



Example: Gamma(4.3, 6.2)

```
## see how samples from chain compare to Gamma density
hist(X, main = "Histogram of MCMC samples", prob = TRUE)
curve(expr = dgamma(x = x, shape = 4.3, rate = 6.2), from = 0, to = 2, add = TRUE, col = "blue")
```



Bayesian Analysis

O-ring Challenger Data

Bayesian Reanalysis of the Challenger O-Ring Data (http://www.calvin.edu/library/Remelts/orings.pdf)

Bayesian logistic regression

- Using a noninformative prior
- let's us use regular logistic parameters with the benefits of Bayesian interpretation

O-ring Challenger Data

The following is a well covered logistic regression example using the O-ring data set related to the 1986 space shuttle Challenger exploision. The output is modeled as the probability of failure (Y = 1) given the data.

$$P(Y=1|X=x) = p = \frac{exp(\alpha + x\beta)}{1 + exp(\alpha + x\beta)}$$

Or, equivalently with the logit transformation on P

$$logit(p) = \frac{p}{1-p} = \alpha + x\beta$$

33/50

O-ring Challenger Data

```
failure <- c(1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
             0, 0, 0, 1, 0, 0, 0, 0, 0)
temperature <- c(53, 57, 58, 63, 66, 67, 67, 67, 68,
                 69, 70, 70, 70, 70, 72, 73, 75, 75,
                 76, 76, 78, 79, 81)

df = data.frame(failure, temperature)
head(df)
```

```
##   failure temperature
## 1       1          53
## 2       1          57
## 3       1          58
## 4       1          63
## 5       0          66
## 6       0          67
```

O-ring Challenger Data

The frequentist logistic regression

```
##
## Call:
## glm(formula = failure ~ temperature, family = binomial(link = "logit"),
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0611  -0.7613  -0.3783   0.4524   2.2175
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  15.0429     7.3786   2.039   0.0415 *
## temperature  -0.2322     0.1082  -2.145   0.0320 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 28.267  on 22  degrees of freedom
## Residual deviance: 20.315  on 21  degrees of freedom
## AIC: 24.315
##
## Number of Fisher Scoring iterations: 5
```

O-ring Challenger Data

Observed are

$$Y_i \sim Bernoulli(p(x_i)),\ \text{where}\ p(x_i) = \frac{exp(\alpha + x_i\beta)}{1 + exp(\alpha + x_i\beta)}$$

where $p(x_i)$ is the probability of O-ring failure at temperature x_i. The likelihood is

$$L(\alpha, \beta|\mathbf{y}) \propto \prod_{i=1}^{n} \left(\frac{exp(\alpha + x_i\beta)}{1 + exp(\alpha + x_i\beta)}\right)^{y_i} \times \left(\frac{1}{1 + exp(\alpha + x_i\beta)}\right)^{1-y_i}$$

and as a prior Robert & Casella choose

$$\pi_\alpha(\alpha|b) \times \pi_\beta(\beta) = \frac{1}{b} e^{\alpha} e^{-e^{\alpha}/b}$$

which puts an exponential prior on $log(\alpha)$ and a flat prior on β (uniform), and insures a proper posterior distribution. Note that priors on α and β are independent. This will be important for computational purposes in the M-H algorithm.

The prior above is an exponential distribution with $Exp(\lambda)$, where $\lambda = \frac{1}{b}e^{\alpha}$

36/50

O-ring Challenger Data

```
## Output from ML estimation from Logistic Regression
## MLEs make great starting valules
a.mle <- as.numeric(fit$coefficients[1])
b.mle <- as.numeric(fit$coefficients[2])
var.a.mle <- summary(fit)$cov.scaled[1, 1]
var.b.mle <- summary(fit)$cov.scaled[2, 2]

b.hyper <- exp(a.mle + 0.577216) ## hyper parameter
## 0.577216 is "Euler’s constant"
```

37/50

O-ring Challenger Data

Let's set up the posterior and proposal distributions

```
## setting up functions

# Posterior distribution
dPosterior <- function(theta, y = failure, x = temperature){
    ## density of Y is binomial/bernoulli
    a <- theta[1]
    b <- theta[2]
    p <- 1 - 1 / (1 + exp(a + b * x)) ## logistic CDF
    lik <- exp(sum(dbinom(y, size=1, prob=p, log=TRUE)))
    dprior <- exp(a) * exp(-exp(a) / b.hyper) * 1/b.hyper ## density of prior
    return(lik * dprior)
}
```

O-ring Challenger Data

```
# Proposal distribution (independent proposal, so "theta0" is not used)
dProposal <- function(theta){
    ## ignore theta0
    a <- theta[1]
    b <- theta[2]
    # a <- log(rexp(1, 1 / b.hyper)) ## remember, log for computational purposes
    # try: exp(rexp(1, 1/b.hyper)) ## Inf
    pr1 <- exp(a) * exp(-exp(a) / b.hyper) * 1/b.hyper
    pr2 <- dnorm(b, b.mle, sqrt(var.b.mle))
    return(pr1 * pr2)
}
```

O-ring Challenger Data

```r
rProposal <- function(theta0){
    ## independent proposals for a and b
    #a <- log(rexp(1, 1 / b.hyper))
    a <- log(rexp(1, 1 / b.hyper)) ## log for computational purposes
    b <- rnorm(1, b.mle, sqrt(var.b.mle))
    return(c(a, b))
}
```

O-ring Challenger Data

Now we run the M-H algorithm

```
## Metropolis-Hastings set up to run
# Run Metropolis-Hastings
N = 1000000
BurnIn =  5000
```

O-ring Challenger Data

Code not shown here. Provided in seperate document.

Uses regular Metropolis Hastings algorithm

42/50

O-ring Challenger Data

```
print("acceptance rate: ")
```

```
## [1] "acceptance rate: "
```

```
print(accept/(N+BurnIn))
```

```
## [1] 0.09512438
```

```
MH.Results <- x[-(1:BurnIn), ]
alphaMH <- MH.Results[,1]
betaMH <- MH.Results[,2]
```

O-ring Challenger Data

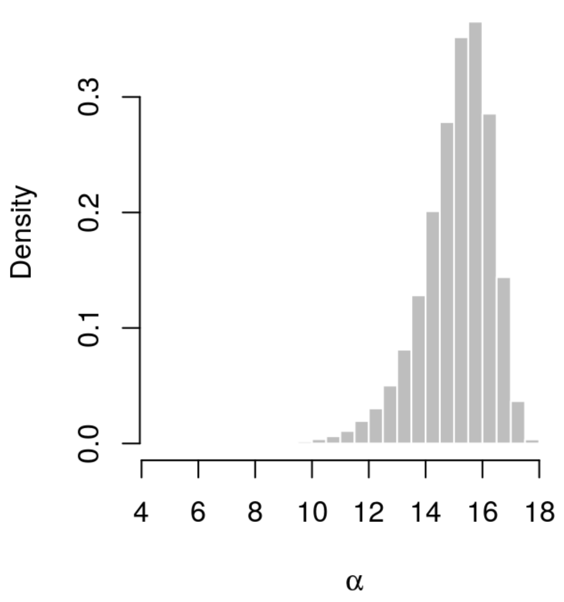
Summary

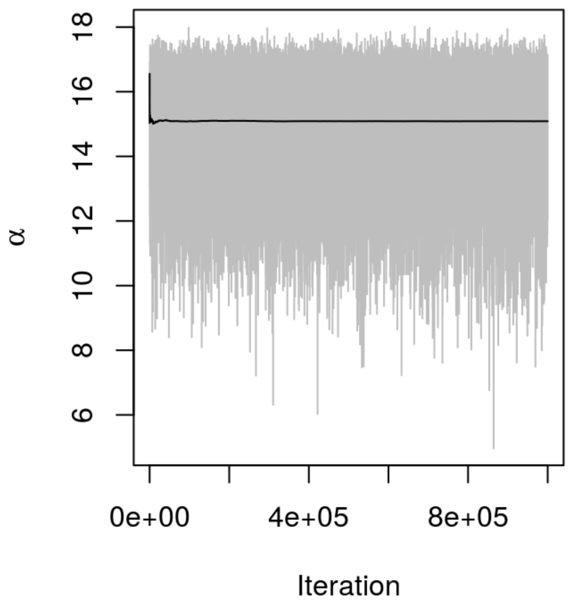
```
summary(MH.Results) ## summary of parameters
```

```
##        V1               V2
##  Min.   : 4.962   Min.   :-0.30365
##  1st Qu.:14.430   1st Qu.:-0.24739
##  Median :15.276   Median :-0.23582
##  Mean   :15.087   Mean   :-0.23372
##  3rd Qu.:15.957   3rd Qu.:-0.22248
##  Max.   :18.012   Max.   :-0.09881
```

44/50

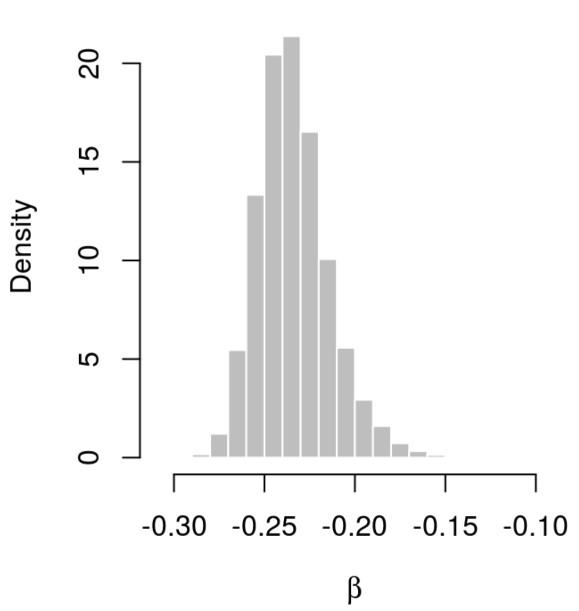
O-ring Challenger Data

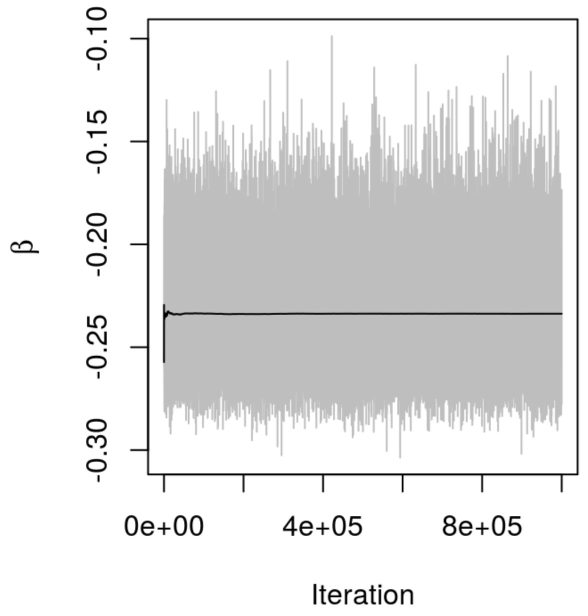




45/50

O-ring Challenger Data





Exercise Student's t density with v degrees of freedom

Calculate the mean of a t distribution with $v = 4$ degrees of freedom using a M-H algorithm with candidate densities $N(0, 1)$ and $t_{v=2}$.

Appendix

Solution to Student's t density with v degrees of freedom

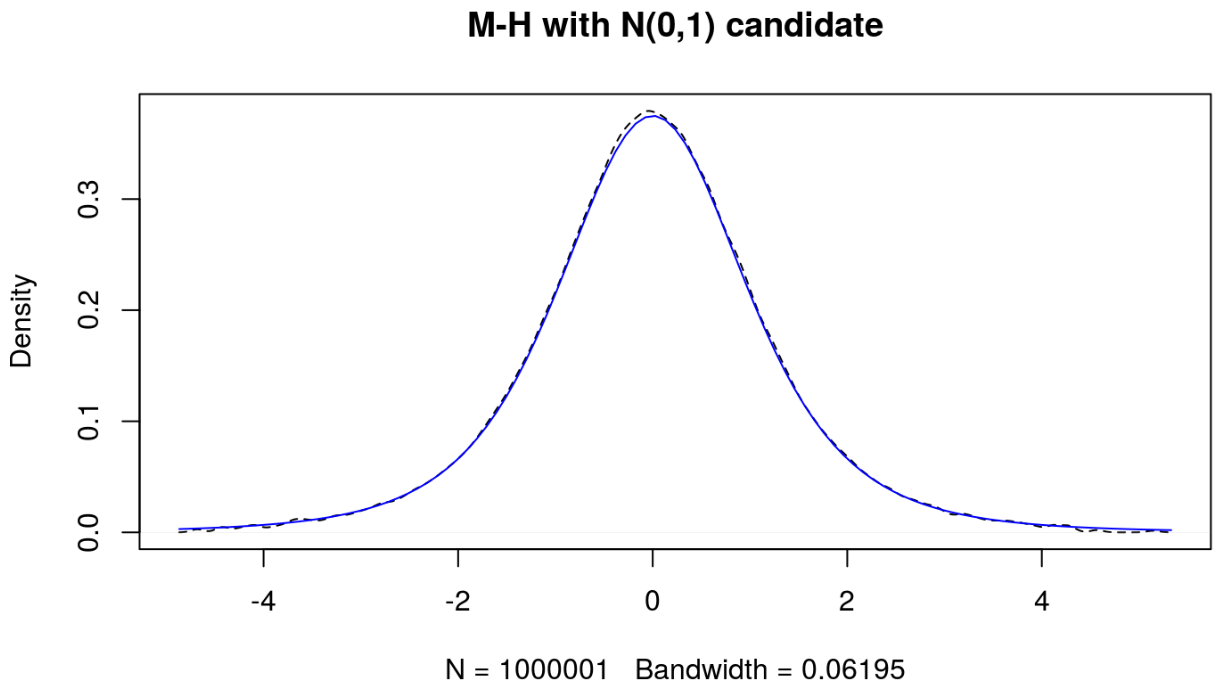
Calculate the mean of a t distribution with $v = 4$ degrees of freedom using a M-H algorithm with candidate densities $N(0,1)$ and $t_{v=2}$.

```
set.seed(987)
N = 10^6

#dt(x = x, df = 4)

X = numeric(N)
X[1] = rnorm(1) ## initialize the starting value

for(i in 1:N){
    Y = rnorm(1) ## independent of X_i
    rho = (dt(Y, df = 4) * dnorm(X[i])) /
            (dt(X[i], df = 4) * dnorm(Y))
    U = runif(1)
    if(U <= rho){
        X[i+1] = Y
    } else{
        X[i+1] = X[i]
    }

}
```

49/50

Solution to Student's t density with v degrees of freedom

```
plot(density(X), type = "l",
        lty = 2, main = "M-H with N(0,1) candidate")
curve(dt(x, df = 4), add = TRUE, col = 4)
```



50/50