Assignment 3

Bjørn Christian Weinbach

6th November, 2020

Clear R environment

```r
rm(list = ls())
```

Problem 1. Random number generation and Monte Carlo integration.

a)

In this problem the task is to sample n independent samples from the probability distribution

$$p(x) = \frac{2^{\frac{1}{4}}\Gamma\left(\frac{3}{4}\right)}{\pi}\exp\left(-\frac{x^4}{2}\right), \qquad -\infty < x < \infty$$

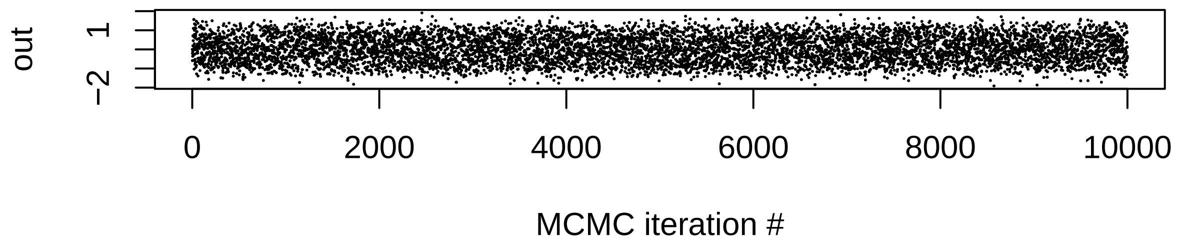
One might wish to implement a regular Metropolis-Hastings random walk for sampling this distribution, but there is one major disadvantage for using this approach: (Wikipedia contributors 2020)

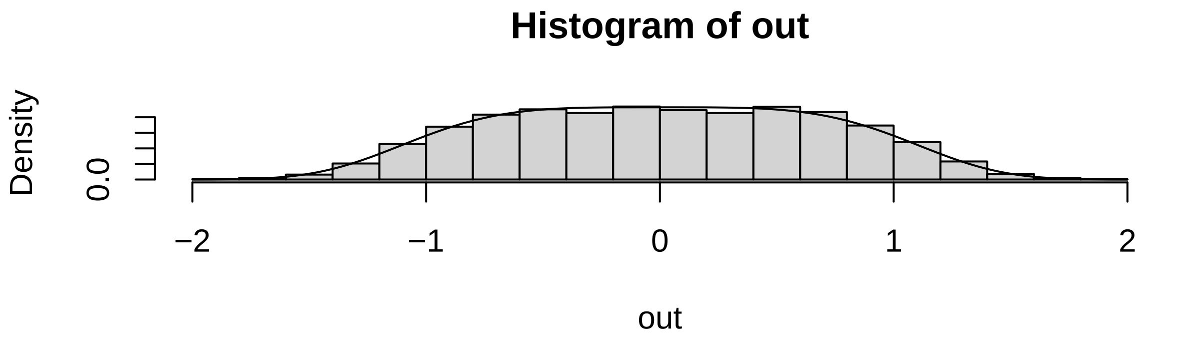
The samples are correlated, even though over the long term they do correctly follow $p(x)$. This is due to the fact that given that you know the current time step, you have a pretty good estimate of what the value of the next one is. In short, the adjacent samples are autocorrelated.

Because of this and the task specifically asking for independent samples. A regular random walk metropolis-hastings approach is not feasible, we then turn to the independent metropolis-hastings approach:

In R:

```r
# general 1d independence metropolis hastings
oneD.IRWMH <- function(prob,
                       sigma=1.0,
                       theta1=0.0,
                       Nsamp=10000){
  res <- numeric(Nsamp)
  # allocate memory
  res[1] <- theta1
  # old importance weight
  wt.old <- prob(res[1])/dnorm(res[1], sd=sigma)
  Nacc <- 0
    for(i in 2:Nsamp){
      # proposal (note, independent of past)
      thetaStar <- rnorm(1, sd=sigma)
      # new importance weight
      wt.star <- prob(thetaStar)/dnorm(thetaStar, sd=sigma)
      # accept probability
      alpha <- min(1.0, wt.star/wt.old)
      # accept/reject
      if(runif(1) < alpha){
        res[i] <- thetaStar
        wt.old <- wt.star
        Nacc <- Nacc+1
      } else {
        res[i] <- res[i-1]
      }
    }
  return(res)
}

# pdf from mandatory pdf
pdf <- function(x) {
  return ((2^0.25*gamma(3/4)/pi) * exp(-x^4 / 2))
}


# sample n samples
n <- 10000
out <- oneD.IRWMH(pdf, theta1 = 0.0, sigma=1, Nsamp = n)

# plot
par(mfrow=c(2,1))
plot(1:length(out),out,pch=20,cex=0.1,xlab="MCMC iteration #")
hist(out, probability = TRUE)
curve(pdf, add = TRUE, -2, 2)
```





b)

Now let's sample from the distribution

$$p(x) = 2x \exp(-x^2), \qquad 0 < x < \infty$$

This function seems to be quite simple to find the CDF for and direct sampling via inverse transform sampling is possible. We get

$$F(x) = \int_0^x p(x) = 1 - e^{-x^2}$$

And by the inverse sampling method, we can calculate

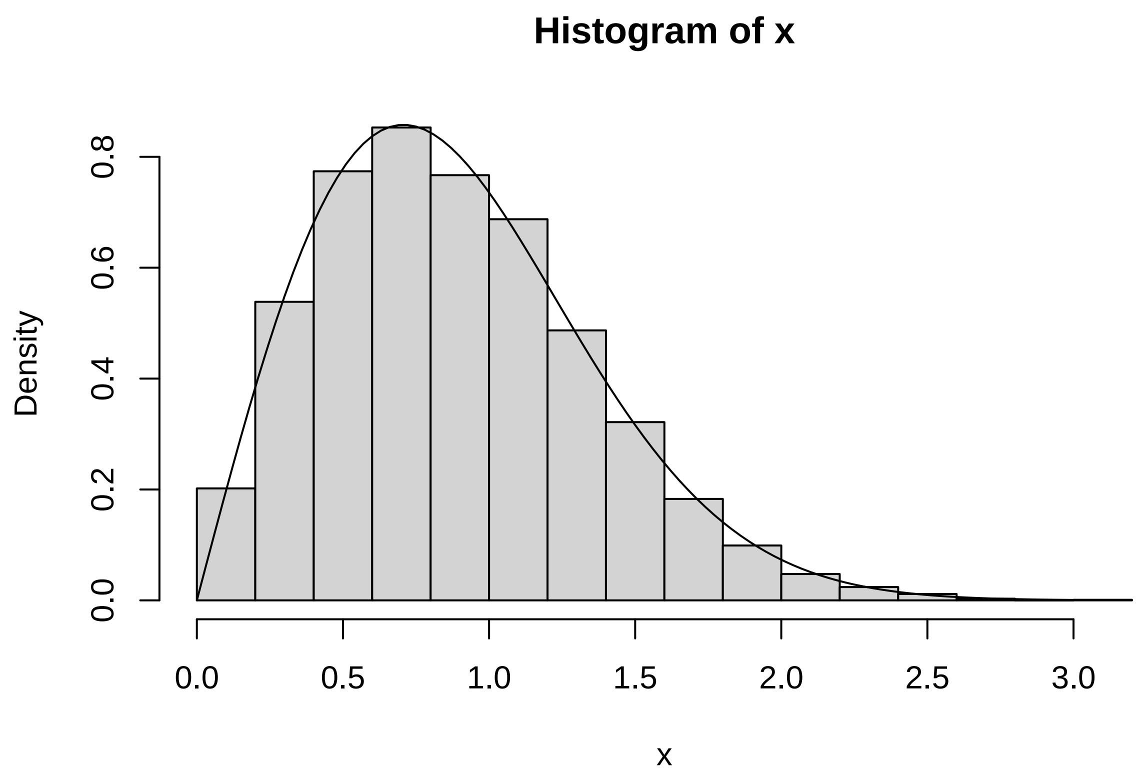
$$X = F^{-1}(U) = \pm\sqrt{-\ln(1-u)}$$

Given the nature of calculating square roots, we get positive and negative X values but since the support of the pdf clearly states that $0 < x < \infty$ we reject the negative values.

in R:

```r
# Use inverse sampling method to sample from
# p(x) = 2x exp(-x^2)
inverse_sampling <- function(Nsamples) {
  # Sample n samples from uniform distribution
  u <- runif(Nsamples)
  # Return X = F^-1(U)
  return(sqrt(-log(1-u)))
}
```

```r
# PDF
pdf2 <- function(x) {
  return(2*x * exp(-x^2))
}

x <- inverse_sampling(10000)
hist(x, probability = TRUE)
curve(pdf2, 0, 3, add = TRUE)
```



c)

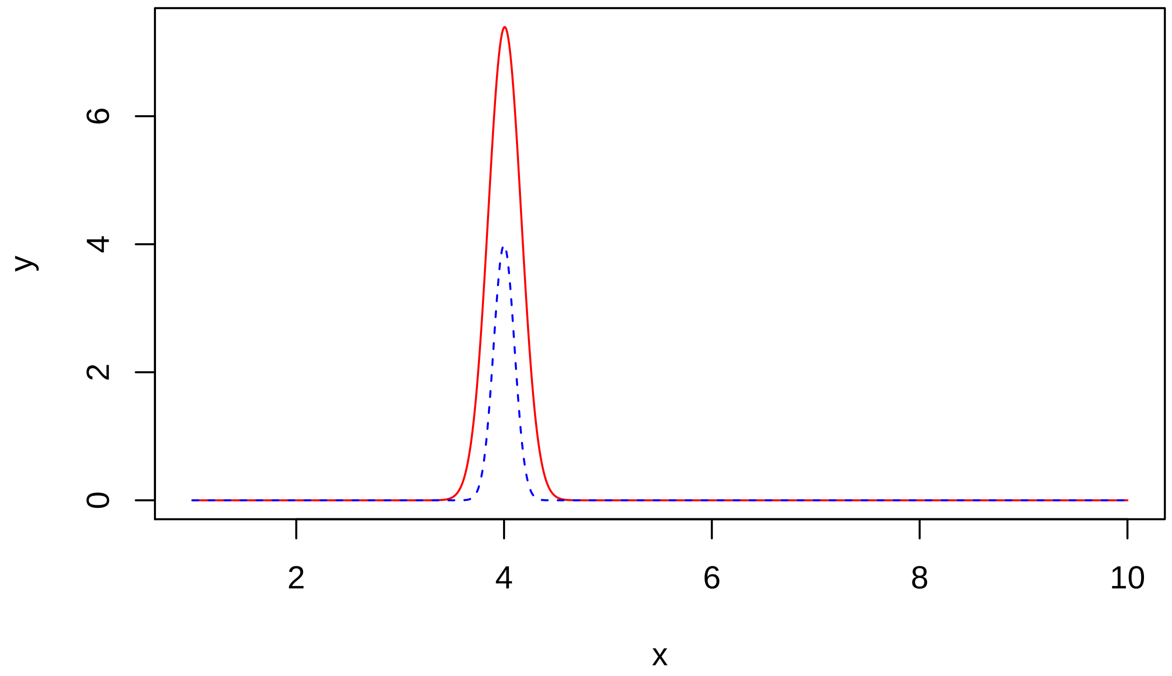
Now we will consider the integral

$$\int_0^\infty \exp(\sqrt{x}) \exp(-20(x-4)^2)\, dx$$

Let's evaluate the integral using importance sampling. First let's plot the function $g(x)$ that is being integrated:

In R:

```r
g <- function(x) {
 exp(sqrt(x))*exp(-20*(x-4)^2)
}

f <- function(x) {
  dnorm(x, mean=4, sd=0.1)
}

x<-seq(1,10,0.01); y1=g(x); y2=f(x)
plot(x, y1, type="l", pch=19, col="red", xlab="x", ylab="y")
lines(x, y2, pch=18, col="blue", type="l", lty=2)
legend(1, 95, legend=c("Line 1", "Line 2"),
       col=c("red", "blue"), lty=1:2, cex=0.8)
```



With the function we want to integrate in $g(x)$ in red and the $f(x)$ in dashed blue. We want to calculate $E\left(\frac{g(x)I(x\in A)}{f(x)}\right)$ by sampling X values from f and calculating the empirical mean of $\frac{g(x)I(x\in A)}{f(x)}$ where I is the indicator function for the support.

```
importance <- function(Nsamp) {
  # Sample from f
  x <- rnorm(Nsamp, 4, 0.1)
  # calculate empirical mean
  return(mean((g(x)*(x>0))/(f(x))))
}

# Estimate of integral using importance sampling
importance(100000)
```

```
## [1] 2.928784
```

```
# Numerical integration
integrate(g, 0, Inf)$value
```

```
## [1] 2.929669
```

Problem 2. Smile shaped target

a)

In this exercise, we shall sample from the distribution

$$\log g(\boldsymbol{\theta}) = -\frac{\theta_1^2}{2} - \frac{(\theta_2 - \theta_1^2)^2}{2}, \qquad \infty < x < \infty$$

To do this, an 2D random walk with multivariate normal proposals as been implemented in R.

In the code below, a 2D random walk with proposal

$$N(\boldsymbol{\theta}, \Sigma)$$

Where

$$\Sigma = \begin{bmatrix} 2, 0 \\ 0, 2 \end{bmatrix}$$

and

$$\boldsymbol{\theta} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Yielded the greatest effective sample size.

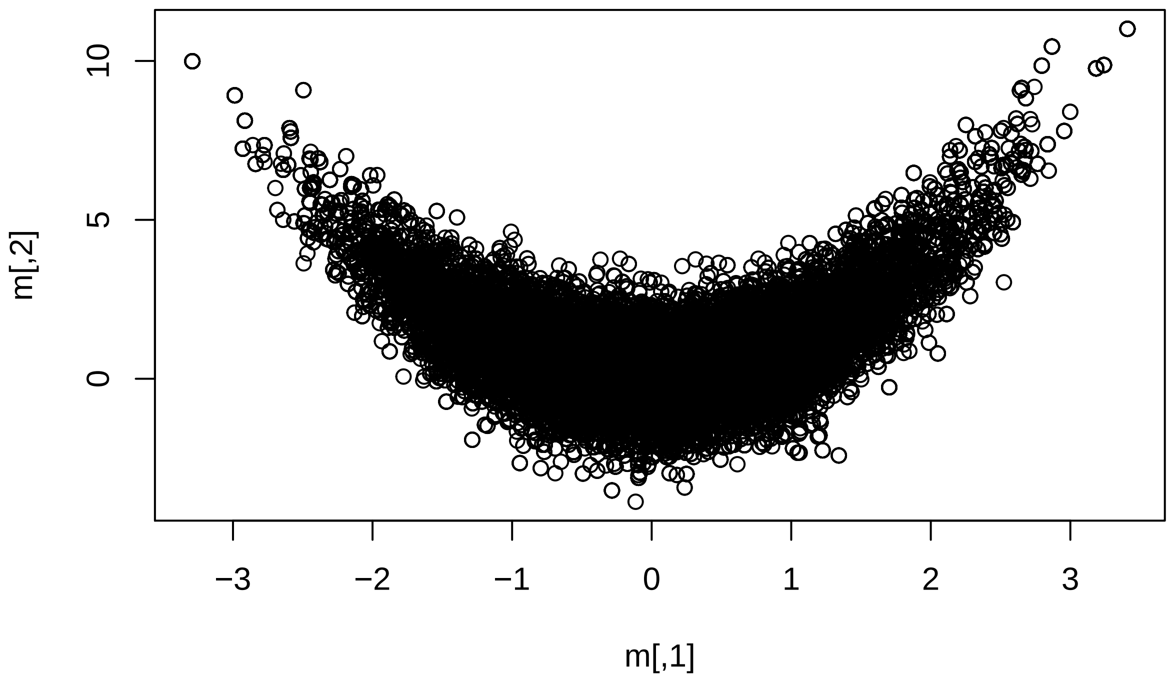
In R:

```r
# Load the multivariate normal library
library(mvtnorm)
library(coda)

# Target function from
logg <- function(theta) {
  return(-theta[1]^2/2 - (theta[2] - theta[1]^2)^2 /2)
}

# 2D Random walk using log g
smile_shaped <- function(logg,
                         sigma=diag(2),
                         theta=c(0.0, 0.0),
                         Nsamp=100000){

  # Reserve 2xNsamp matrix with zeros
  res <- matrix(0, Nsamp, 2)

  # Set initial conditions
  res[1,] <- theta

  # Calculate old log-probability
  logold <- logg(theta)

  # accept counter
  Nacc <- 0
  # Iterate no of samples.
  for(i in 2:Nsamp){
    # Proposal step
    new <- res[i-1,] + rmvnorm(1, theta, sigma)
    # Log-p of proposal step
    lognew <- logg(new)
    # Evaluate step
    logfrac <- exp(min(0.0,lognew-logold))
    # Accept or reject new step
    if(runif(1)<logfrac && is.finite(logfrac)){
      # accept
      res[i,] <- new
      logold <- lognew
      Nacc <- Nacc+1
    } else {
      # reject
      res[i,] <- res[i-1,]
    }
  }
  print(paste("Accept prob: ", Nacc/Nsamp))
  return(res[5000:Nsamp,])
}

# Covariance matrix
sigma <- matrix(c(4.5, 0, 0, 4.5), nrow=2, ncol=2)
m <- smile_shaped(logg=logg, sigma=sigma, c(0.0, 0.0), 50000)
```

```
## [1] "Accept prob:  0.23232"
```

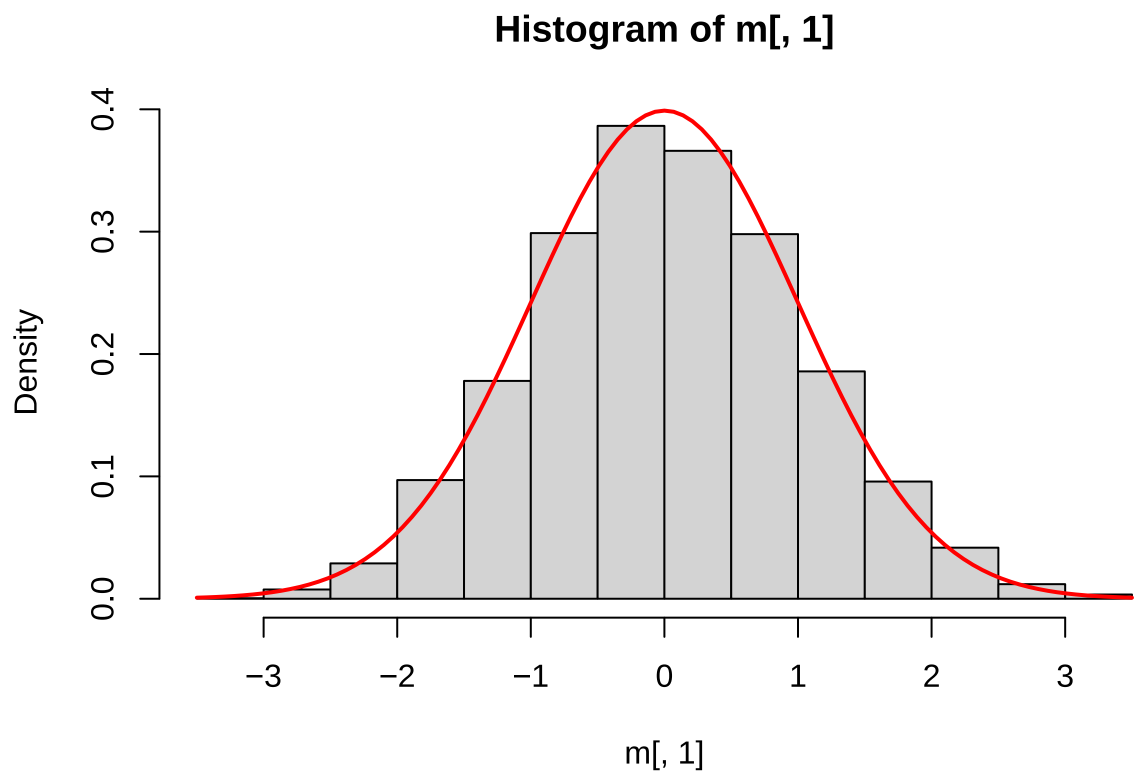
Let's plot the "Smile shaped" distribution:

```r
# Plot multivariate distribution
plot(m)
```



Let’s also see if the marginal distribution of θ_1 is standard normal:

```
# Plot marginal of theta1 and compare with standard normal
hist(m[,1], probability=TRUE)
curve(dnorm(x, mean=0, sd=1),
      col="red", lwd=2, add=TRUE, yaxt="n")
```



To check if we have explored the distribution well, let's calculate the effective sample size of $theta_1$ and $theta_2$.

```r
ESS <- function(x) {
  coda::effectiveSize(x)
}

ESS(m[,1])
```

```
##     var1
## 3350.318
```

```r
ESS(m[,2])
```

```
##     var1
## 1891.734
```

Problem 3. IMH for simple logistic regression problem

```r
# Load the data set
df <- data.frame(read.table("logistic_regression_data.txt"))
x <- df$x
y <- df$y

# function returning a log-posterior kernel for theta
logistic.lp <- function(theta) {
  alpha <- theta[1]
  beta <- theta[2]
  # log-likelihood
  Eeta <- exp(alpha + beta*x)
  p <- Eeta/(1.0 + Eeta)
  log.like <- sum(dbinom(y, size=1, prob = p, log=TRUE))

  # priors
  log.prior <- dnorm(alpha, sd=10, log=TRUE) + dnorm(beta, sd=10, log=TRUE)

  # log-posterior kernel
  return(log.like+log.prior)
}
```

a) Sample α and β using metropolis-hasting and bayesian inference

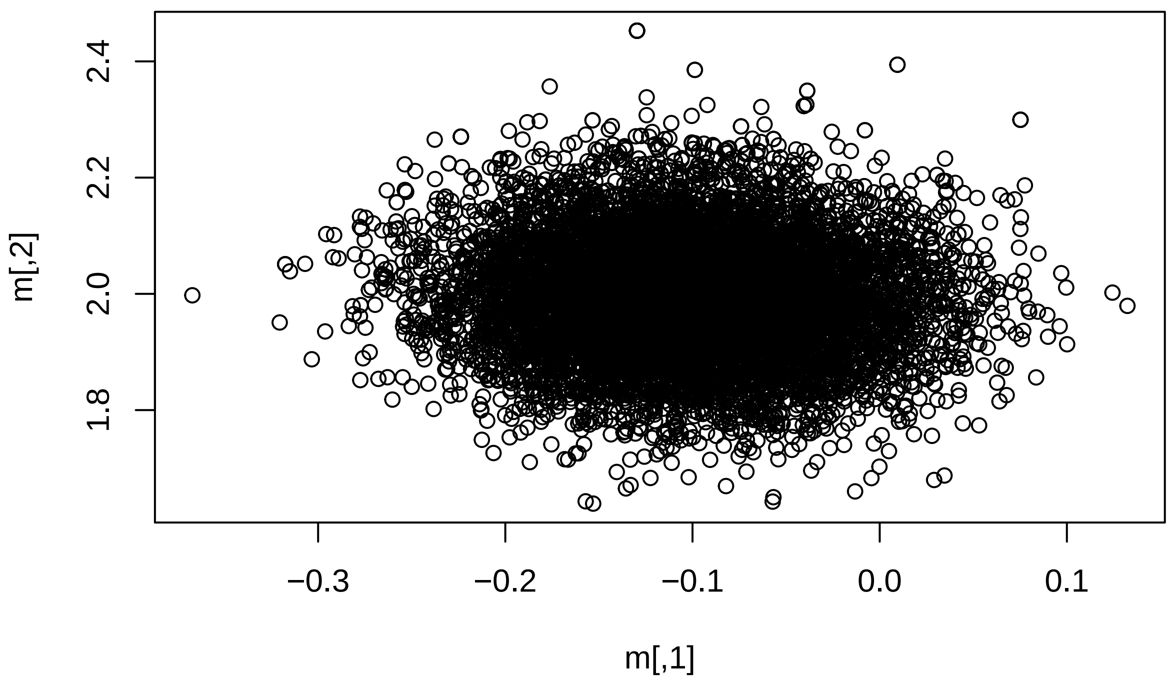
In this task, we will use independent metropolis-hastings sampling for the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{y})$ using a $N(\hat{\theta}, \delta\hat{\Sigma})$ proposal distribution.

In R:

```r
# Independent sampler
IndepMH <- function(lprob, theta=c(0, 0), sigma=diag(2), Nsamp=1000, S=1){
  # Allocate space
  res <- matrix(0, Nsamp, 2)
  # Set initial values
  res[1,] <- theta
  # Old importance
  pold <- lprob(theta) - dmvnorm(theta, mean=theta ,sigma=sigma, log=TRUE)
  # No of accept
  Nacc <- 0
  for (i in 2:Nsamp){
    # Sample theta from multivariate normal with scaled sigma
    new <- rmvnorm(1, theta, S*sigma)
    # Importance of new theta
    pnew <- lprob(new) - dmvnorm(new, mean=theta, sigma=sigma, log=TRUE)
    frac <- exp(min(0.0, pnew-pold))
    if(runif(1)<frac && is.finite(frac)){
      # Accept
      res[i,] <- new
      pold <- pnew
      Nacc <- Nacc+1
    } else {
      # Reject
      res[i,] <- res[(i-1),]
    }
  }
  print(paste0("accept rate : ",Nacc/Nsamp))
  return(res[1000:Nsamp,])
}

sigma <- matrix(c(0.00653,-0.00058,-0.00058,0.01689),2,2)

m <- IndepMH(logistic.lp, c(-0.102, 1.993), sigma, 10000, 0.6)
```

```
## [1] "accept rate : 0.982"
```

```r
plot(m)
```



Calculate ESS

```r
ESS(m[,1])
```

```
##     var1
## 8326.297
```

```r
ESS(m[,2])
```

```
##     var1
## 8141.387
```

b/c) Plot quantiles of data and find $x*$ such that $P(m(x*) > 0.8) = 0.99$

To solve this problem, we calculate $m(x*)$ the median and quantiles for $x \in [-5, 5]$ where α and β has been estimated based on the dataset provided in the mandatory assignment.

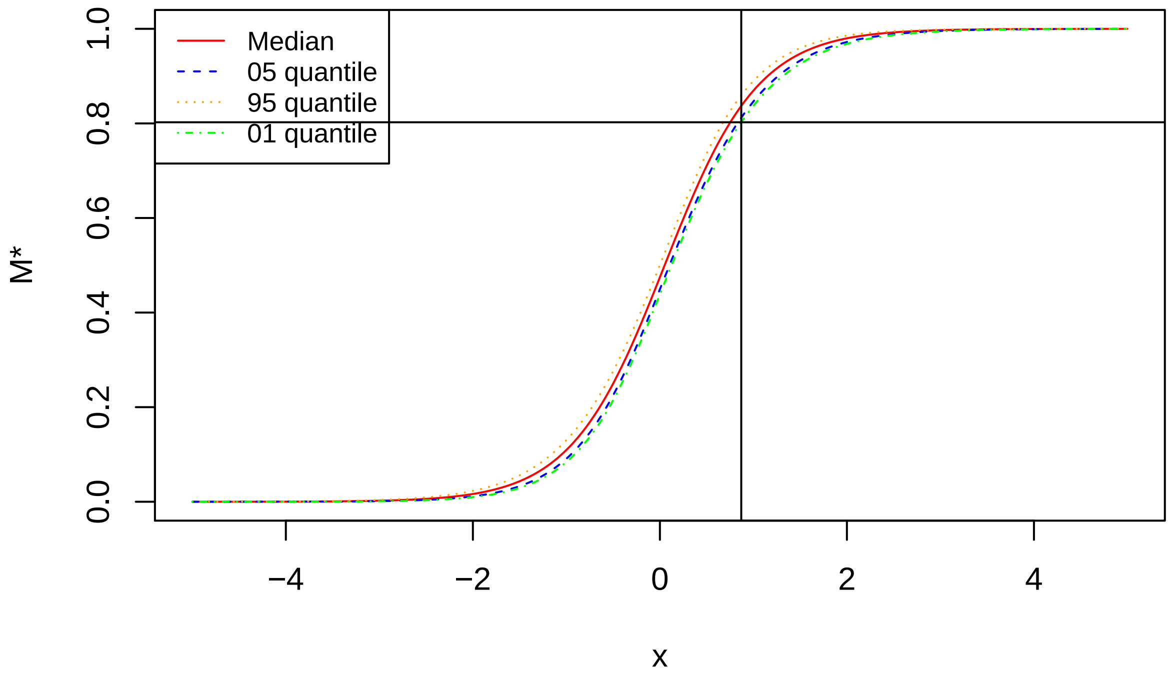
We use the following algorithm to solve this problem:

1. Given data, estimate α, β using independent metropolis hastings
2. Select sequence of $x*$ we want to explore
3. Iterate through each value of $x*$
4. Calculate median and 01th, 05th and 95th quantiles and return these as a vector
5. Store vector as row in matrix
6. Repeat steps 4-5 for each value in $x*$

When this algorithm terminates, we have a 2D matrix of medians and quantiles for all $x*$. We use this for plotting and finding the $x*$ where $P(m(x*) > 0.8) = 0.99$

In R:

```r
# Plotdist is function for calculating values for plotting
plotdist <- function (x) {
  # Calculate m(x*)
  mstar <- exp(m[,1] + m[,2]*x) / (1 + exp(m[,1] + m[,2]*x))
  # calulcate median given x*
  med <- median(mstar)
  # Caltulate quantiles given x*
  quant05 <- quantile(mstar, 0.05)
  quant95 <- quantile(mstar, 0.95)
  quant01 <- quantile(mstar, 0.01)
  # return vector of calculations
  return(c(med, quant05, quant95, quant01))
}

# sequence of x* values
x <- seq(-5, 5, 0.01)
# plot
plotmat <- matrix(0, length(x), 4)
# set index
i <- 0
# Loop through all x* values in sequence
for (val in x) {
  # increment index for each value in xÆ
  i <- i+1
  plotmat[i,] <- plotdist(val)
}

# First value larger than 0.8 (for plotting horizontal)
horizontal <- plotmat[, 4][plotmat[, 4] > 0.8][1]

# First x-value of m-value with 99% probability of being larger than 0.8
vertical <- x[Position(function(x) x > 0.8, plotmat[, 4])]

# Plot all medians and quantiles conditional on x*
plot(x, plotmat[, 1], type="l", col="red", ylab="M*")
lines(x, plotmat[, 2], type="l", col="blue", lty=2)
lines(x, plotmat[, 3], type="l", col="orange", lty=3)
lines(x, plotmat[, 4], type="l", col="green", lty=4)
# Add vertical line for x values
abline(v = vertical, col="black")
# add horizontal line for m values
abline(h = horizontal, col="black")
legend("topleft", legend=c("Median", "05 quantile", "95 quantile", "01 quantile"),
       col=c("red", "blue", "orange", "green"), lty=1:4, cex=0.8)
```



```r
# X* where P(M(X*) > 0.8) = 0.99
print(vertical)
```

```
## [1] 0.87
```

Problem 4. Gibbs sampler for simple linear regression model

a) Find conditional posteriors.

The posterior for τ is given as

$$\pi(\tau|\alpha,\beta,\boldsymbol{y}) \sim \text{gamma}\left(\frac{n}{2}+1, \frac{1}{\frac{1}{2}\sum_{i=1}^{n}(y_i-\alpha-\beta x_i)^2+1}\right)$$

To calculate the posterior of α we see by inspection that the log-density kernel is possibly on the form $\log\pi(\alpha|\beta,\tau,\boldsymbol{y}) = a + b\alpha + c\alpha^2$ which tells us the distribution is normal and that the constant a is unimportant.

Let's calculate $\log\pi(\alpha|\beta,\tau,\boldsymbol{y})$.

From the log-density kernel in the assignment description, we get

$$\log\pi(\alpha|\beta,\tau,\boldsymbol{y}) = c'_\alpha - \frac{\tau}{2}\sum_{i=1}^{n}(y_i-\alpha-\beta x_i)^2 - \frac{\alpha^2}{200}.$$

Where c'_α is a unimportant constant containing all values not dependent on α. By squaring the term in the sum and adding unimportant values to c'_α we now get.

$$\log \pi(\alpha|\beta,\tau,\boldsymbol{y}) = c'_\alpha - \frac{\tau}{2}(n\alpha^2 + \alpha\sum_{i=1}^{n}(2\beta x_i - 2y_i)) - \frac{\alpha^2}{200}$$

By multiplying the τ term and reshuffling we get

$$\log \pi(\alpha|\beta,\tau,\boldsymbol{y}) = c_\alpha - \tau\sum_{i=1}^{n}(\beta x_i - y_i)\alpha - \left(\frac{\tau n}{2} + \frac{1}{200}\right)\alpha^2$$

And we have

$$b_\alpha = -\tau\sum_{i=1}^{n}(\beta x_i - y_i)$$

and

$$c_\alpha = -\left(\frac{\tau n}{2} + \frac{1}{200}\right)$$

and the posterior (from the lecture notes) is

$$\pi(\alpha|\beta,\tau,\boldsymbol{y}) \sim N\left(-\frac{b_\alpha}{2c_\alpha}, -\frac{1}{2c_\alpha}\right)$$

And we do the same for $\pi(\beta|\alpha,\tau,\boldsymbol{y})$

From the log-density kernel in the assignment description, we get

$$\log \pi(\beta|\alpha,\tau,\boldsymbol{y}) = c'_\beta - \frac{\tau}{2}\sum_{i=1}^{n}(y_i - \alpha - \beta x_i)^2 - \frac{\beta^2}{200}.$$

Where c'_β is a unimportant constant containing all values not dependent on β. By squaring the term in the sum and adding unimportant values to c'_β we now get.

$$\log \pi(\beta|\alpha,\tau,\boldsymbol{y}) = c'_\beta - \frac{\tau}{2}\sum_{i=1}^{n}\left(2\alpha\beta x_i + \beta^2 x_i^2 - 2\beta x_i y_i\right) - \frac{\beta^2}{200}$$

By multiplying the τ term and reshuffling we get

$$\log \pi(\beta|\alpha,\tau,\boldsymbol{y}) = c'_\beta - \tau\left(\sum_{i=1}^{n}\alpha x_i - \sum_{i=1}^{n}x_i y_i\right)\beta - \left(\tau\sum_{i=1}^{n}x_i^2 + \frac{1}{100}\right)\beta^2$$

And we have

$$b_\beta = -\tau\left(\sum_{i=1}^{n}\alpha x_i - \sum_{i=1}^{n}x_i y_i\right)$$

and

$$c_\beta = -\left(\tau\sum_{i=1}^{n}x_i^2 + \frac{1}{100}\right)$$

and the posterior (from the lecture notes) is

$$\pi(\beta|\alpha,\tau,\boldsymbol{y}) \sim N\left(-\frac{b_\beta}{2c_\beta}, -\frac{1}{2c_\beta}\right)$$

b) Gibbs sampler with three blocks

Load the data

```r
# Load regression data
df <- data.frame(read.table(file="linear_regression_data.txt"))
x <- df$x
y <- df$y
```

Let's implement the gibbs sampler

```r
gibbs <- function(x, y, theta=c(1.0, -1.2, 1.0), Nsamp=10000) {
  # Result matrix (State stored thrice each iteration)
  res <- matrix(0.0, 3*Nsamp+1, 3)
  # Set state 1 to initial values
  res[1, ] <- theta
  # Set gamma shape parameter
  gammashape <- length(x)/2 + 1
  # Set counter
  k <- 2
  # Iterate...
  for(i in 1:Nsamp){

      # Block 1. Alpha
      balpha <- -theta[3]*sum(theta[2]*x - y)
      calpha <- -theta[3]*length(y)*0.5 - 1/200
      theta[1] <- rnorm(1, -balpha/(2*calpha), -1/(2*calpha))
      # store state
      res[k, ] <- theta
      k <- k+1

      # Block 2. Beta
      bbeta <- -theta[3]*(sum(x*theta[1]) - sum(x*y))
      cbeta <- -theta[3]*0.5*sum(x^2) - 1/200
      theta[2] <- rnorm(1, -bbeta/(2*cbeta), -1/(2*cbeta))
      # store state
      res[k, ] <- theta
      k <- k+1

       # Block 3. Tau
      gammascale <- 1/(0.5*sum((y - theta[1] - theta[2]*x)^2))
      theta[3] <- rgamma(1, gammashape, scale = gammascale)
      # store state
      res[k, ] <- theta
      k <- k+1
  }
  # Return results minus burn in
  return(res[500:k-1,])
}

# Get samples to matrix
m <- gibbs(x, y)
```

```r
# Calculate column means
colMeans(m)
```

```
## [1]  1.088497 -1.206630  1.122432
```

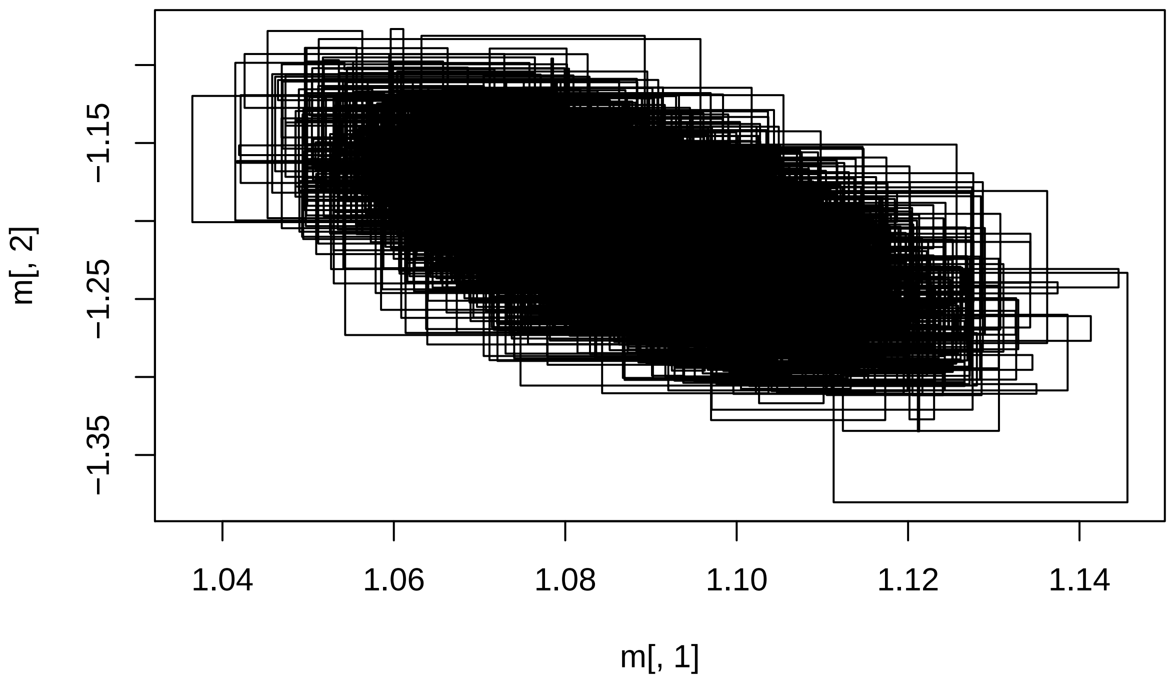
```r
# Compre with one simple linear regression
lm(y~x, data=df)
```

```
##
## Call:
## lm(formula = y ~ x, data = df)
##
## Coefficients:
## (Intercept)            x
##       1.089       -1.207
```

c)

Our means seem pretty good, let's plot trace plots of alpha and beta.

```r
# Traceplot alpha vs beta
plot(m[,1], m[,2], type="l")
```



Also, let's calculate all the effective sample sizes for each variable

```r
ESS(m[,1])
```

```
##     var1
## 5191.684
```

```r
ESS(m[,2])
```

```
##     var1
## 5402.595
```

```r
ESS(m[,3])
```

```
##     var1
## 9983.042
```

We see that we have a pretty good sample size. Now let's investigate whether we can improve this by reducing the no of blocks and making blocks less dependent.

d)

Let's combine the blocks for α and β from two normal distribution to one block of a multinormal distribution. With a posterior distribution $\pi(\alpha, \beta|\tau, \boldsymbol{y}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

where

$$\boldsymbol{\Sigma} = -\boldsymbol{c}^{-1}$$

and

$$\boldsymbol{\mu} = -\boldsymbol{c}^{-1}\boldsymbol{b}$$

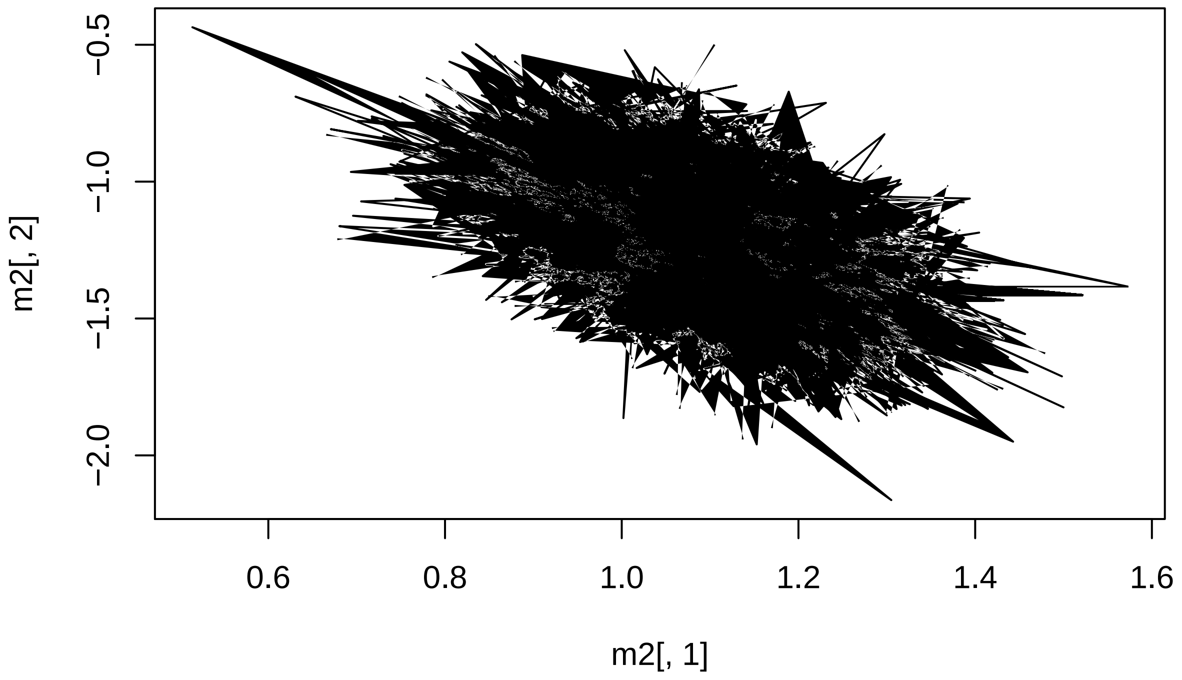
and $\boldsymbol{c}$ and $\boldsymbol{b}$ is given in the assignment.

Let's alter the function for gibbs sampling to have two blocks instead.

```r
gibbs_multi <- function(x, y, theta=c(1.0, -1.2, 1.0), Nsamp=10000) {
  # Result matrix (State stored thrice each iteration)
  res <- matrix(0.0, 2*Nsamp+1, 3)
  # Set state 1 to initial values
  res[1, ] <- theta
  # Set gamma shape parameter
  gammashape <- length(x)/2 + 1
  # Set counter
  k <- 2

  # Calculations that can be done once for improvment of performance
  # Sum of x
  sumx <- sum(x)
  # Sum of x^2
  sumx2 <- sum(x^2)
  # Sum of y
  sumy <- sum(y)
  # Sum of xy
  sumyx <- sum(x*y)
  # No of datapoints
  n <- length(x)

  # Iterate...
  for(i in 1:Nsamp){

    # Block 1. [alpha, beta]
    # C from assignment description
    c <- matrix(c(-n*theta[3] - 0.01, -theta[3]*sumx,
                  -theta[3]*sumx, -theta[3]*sumx2 - 0.01), nrow=2, ncol=2)
    # b from assignment description
    b <- c(theta[3]*sumy, theta[3]*sumyx)

    theta[1:2] <- rmvnorm(1, -solve(c)%*%b, -solve(c))
    res[k, ] <- theta
    k <- k+1

     # Block 2. Tau
    gammascale <- 1/(0.5*sum((y - theta[1] - theta[2]*x)^2))
    theta[3] <- rgamma(1, gammashape, scale = gammascale)
    # store state
    res[k, ] <- theta
    k <- k+1
  }
  # Return results minus burn in
  return(res[500:k-1,])
}

m2 <- gibbs_multi(x, y)
colMeans(m2)
```

```
## [1]  1.089562 -1.207480  1.097435
```

```
plot(m2[,1], m2[,2], type = "l")
```



Now, is this gibbs sampler better than our first implementation? Let's check the effective sample size:

```r
ESS(m2[,1])
```

```
##     var1
## 10845.96
```

```r
ESS(m2[,2])
```

```
##     var1
## 9867.417
```

```r
ESS(m2[,3])
```

```
##     var1
## 9111.711
```

Which makes sense, we sampled *beta* and *alpha* with normal distributions dependent on the values of one another, in this case we sample both from a multinormal distribution according to the given covariance matrix where each pair of samples are independent from one another.

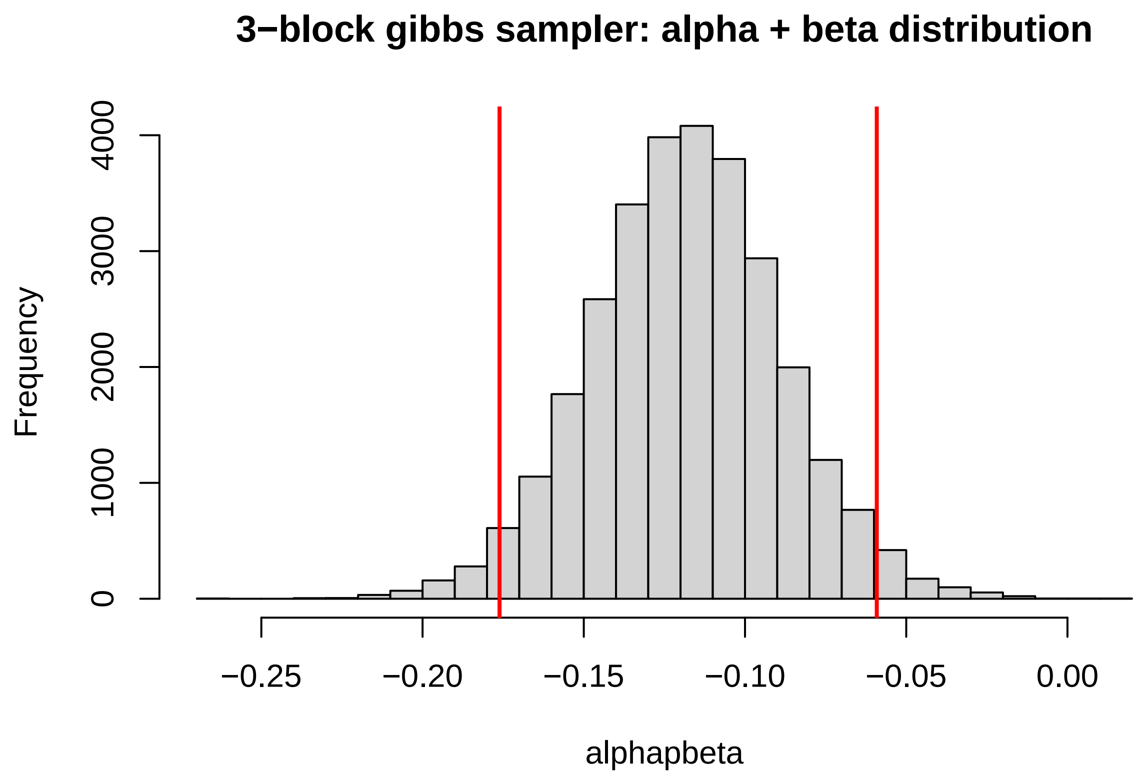
Fewer blocks and striving for independence among blocks will, according to the lectures, give better samples.

e)

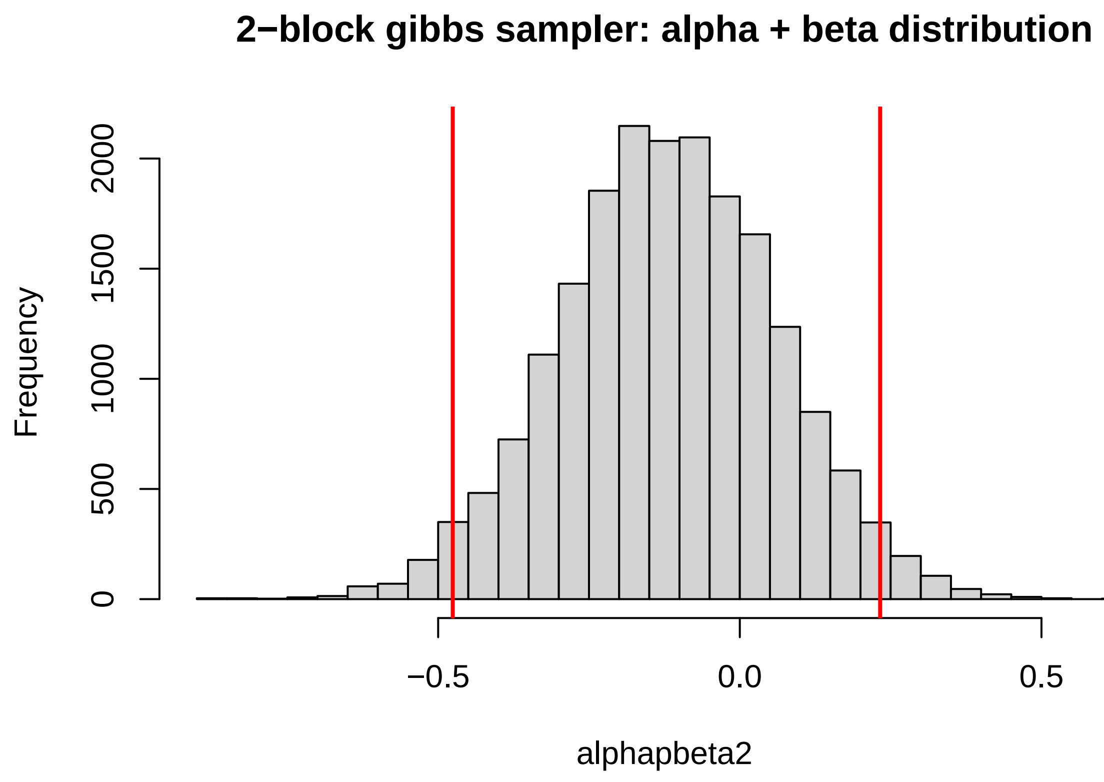
To do a simple "hypothesis test" using quantiles and our samples. for every sample, calculate $\alpha + \beta$ and store the values, calculate the 95% quantile confidence interval and plot the histogram with the interval.

This could be done more appropriately by using a more analytical approach for our calculations but we resort to this simple case.

```r
alphapbeta <- m[,1] + m[,2]
hist(alphapbeta, breaks=25,
     main = "3-block gibbs sampler: alpha + beta distribution")
abline(v = quantile(alphapbeta, 0.975), col="red", lwd="2")
abline(v = quantile(alphapbeta, 1-0.975), col="red", lwd="2")
```



```
alphapbeta2 <- m2[,1] + m2[,2]
hist(alphapbeta2, breaks=25,
     main = "2-block gibbs sampler: alpha + beta distribution")
abline(v = quantile(alphapbeta2, 0.975), col="red", lwd="2")
abline(v = quantile(alphapbeta2, 1-0.975), col="red", lwd="2")
```



In the first distribution, we accept (with a significance of 5%), that $\alpha + \beta \neq 0$.

In the second distribution, we fail to reject the null-hypothesis that
$\alpha + \beta = 0$.

Problem 5

Bibliography

Wikipedia contributors. 2020. “Metropolis-Hastings Algorithm — Wikipedia, the Free Encyclopedia.” https://en.wikipedia.org/w/index.php?title=Metropolis%E2%80%93Hastings_algorithm&oldid=983825650.