Assignment 3

Bjørn Christian Weinbach

4th November, 2020

Clear R environment

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

Problem 1 Random number generation and Monte Carlo integration.

a)

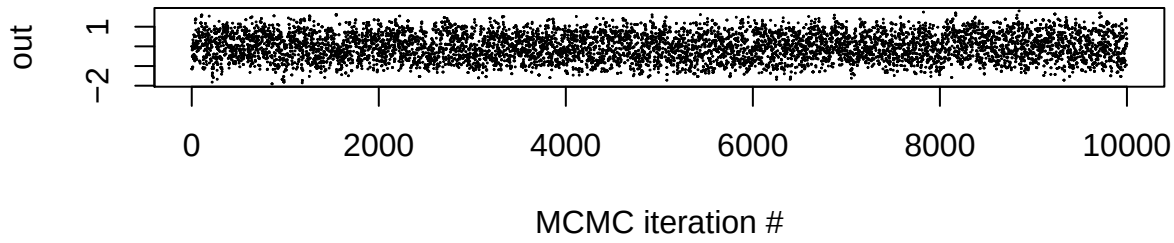
In this problem the task is to sample n independent samples from the probability distribution

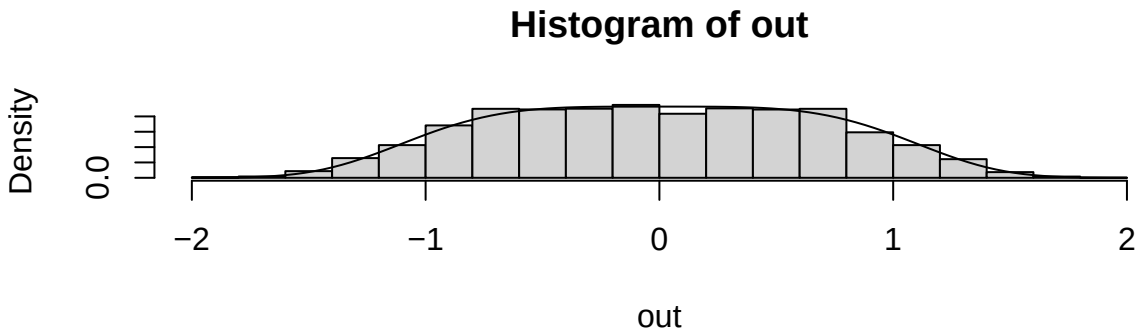
$$p(x) = \frac{2^{\frac{1}{4}}\Gamma\left(\frac{3}{4}\right)}{\pi} \exp\left(-\frac{x^4}{2}\right), \qquad -\infty < x < \infty$$

For this, we can use random walk metropolis hasting. We import the function from the lectures.

```
# general 1d Gaussian proposal random walk MH
oneD.RWMH <- function(lprob, #notice log-density kernel!
                      sigma=1.0,
                      theta1=0.0,
                      n.iter=10000){
  # space for output
  output <- numeric(n.iter)
  # first iterate given
  output[1] <- theta1

  # main iteration loop
  for(t in 2:n.iter){
    # proposal
    thetaStar <- output[t-1] + rnorm(1,sd=sigma)
    # accept probability, for numerical stability we compute
    # the log-accept prob, and then take exp
    alpha <- exp(min(0.0,lprob(thetaStar)-lprob(output[t-1])))
    # accept/reject step
    if(runif(1)<alpha && is.finite(alpha)){
      output[t] <- thetaStar
    } else {
      output[t] <- output[t-1]
    }
  }
  return(output)
}
```

```r
# Logartihm of pdf (needed for metropolis hastings from slides)
logpdf <- function(x){
  c <- log((2^0.25 * gamma(3/4)) / pi)
  return(c - x^4/2)
}

# 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.RWMH(logpdf, theta1 = 0.0, sigma=1, n.iter = 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)
```





We choose this algorithms since it is difficult to integrate the pdf to find the cdf for inverse sampling, but the logarithm of the pdf is quite simple and metropolis hastings random walk is quite effective in low dimensions and for distributions that are difficult to sample directly.

From the histogram where we plot the pdf we see that the sampling is quite effective. We also see from the plot that the random walk explores the distribution space quite nicely.

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

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

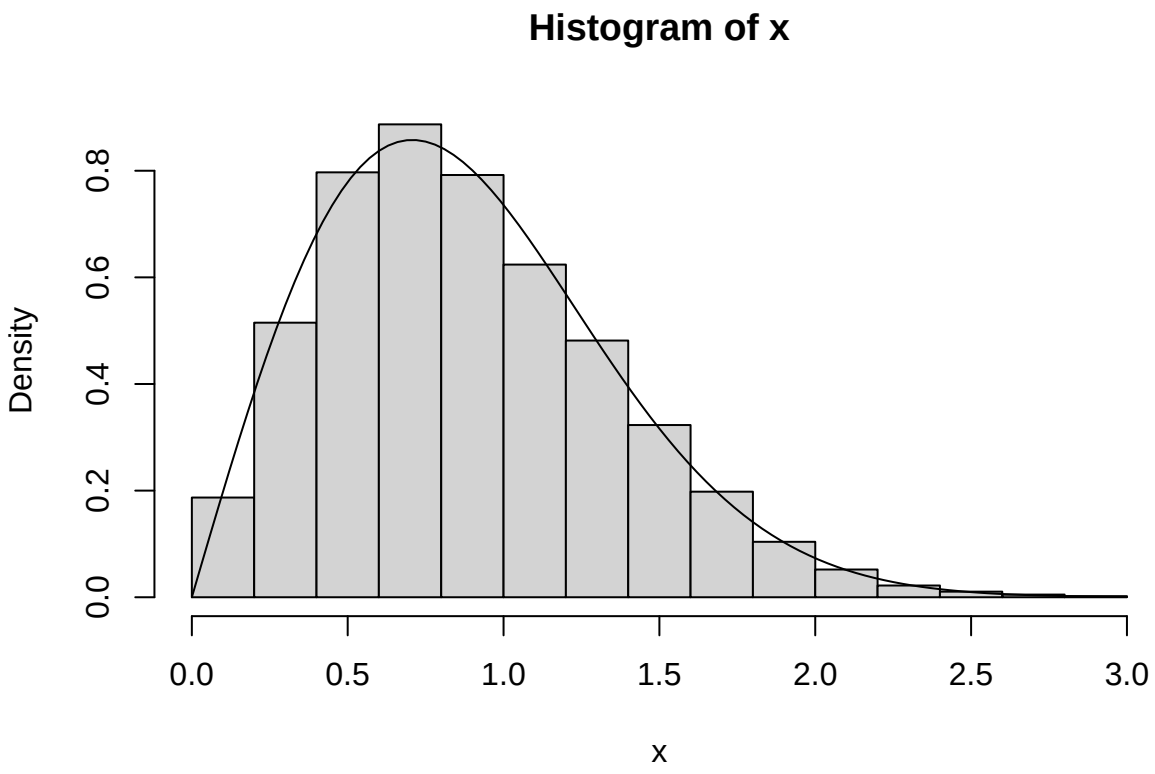
And by the inverse sampling method, we can calculate

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

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)))
}

# 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)

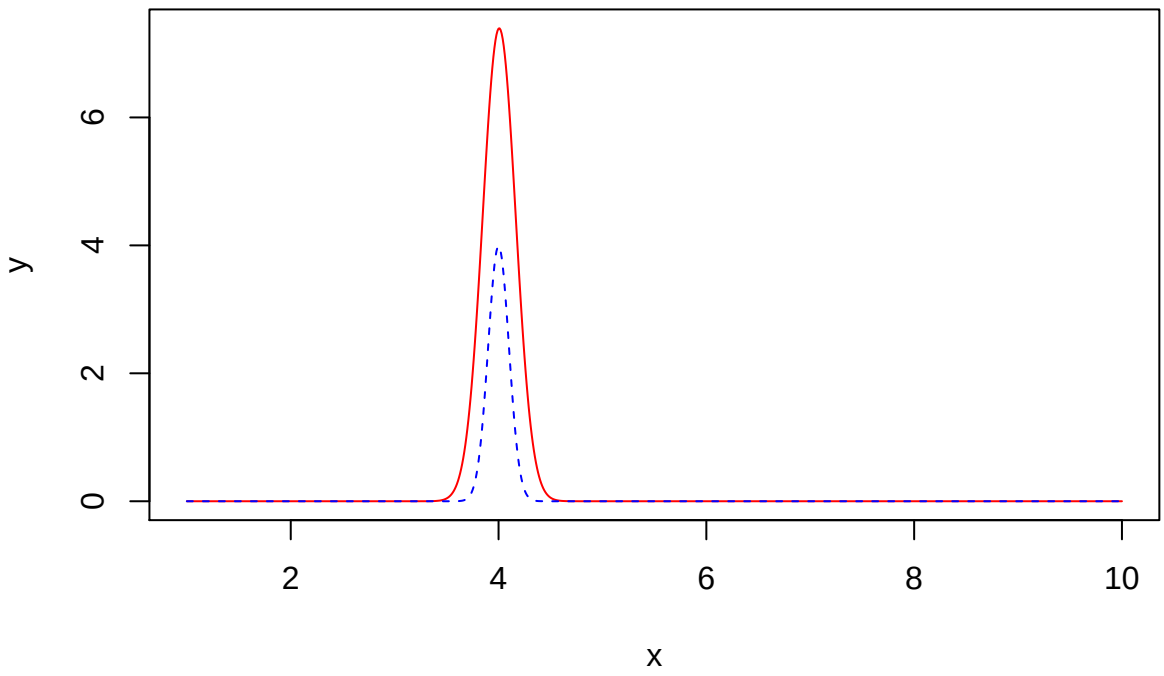
Now we will consider the integral

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

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

In R:

```
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] 2.92655
```

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

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

Problem 2. Smile shaped target

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:

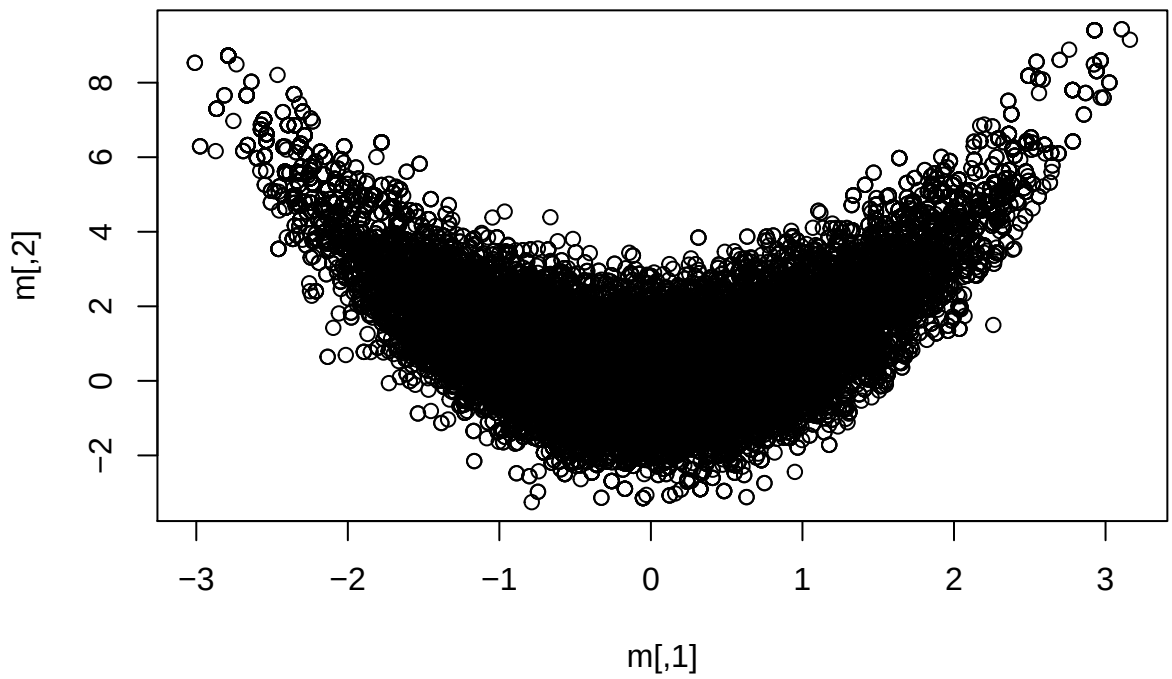
```
# Load the multivariate normal library
library(mvtnorm)
library(coda)

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

```
## [1] "Accept prob:  0.35694"
```

Let's plot the "Smile shaped" distribution:

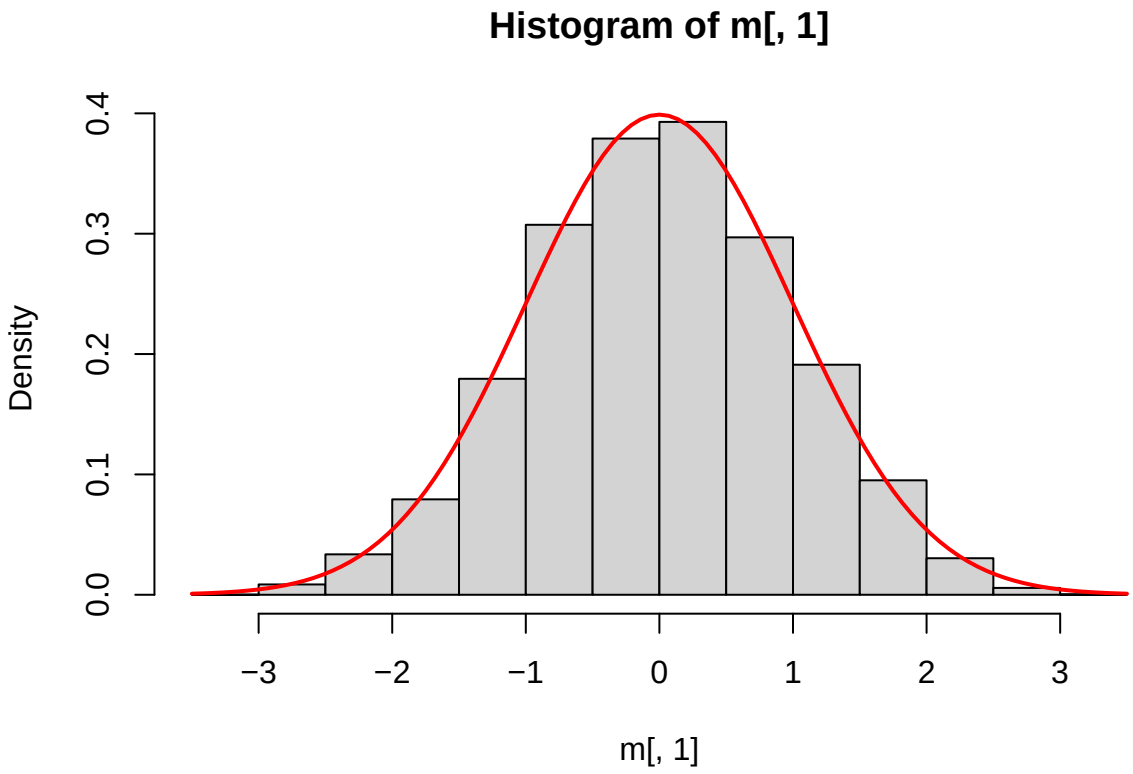
```
# Plot multivariate distribution
plot(m)
```



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 $theta_2$.

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

ESS(m[,1])
```

```
##     var1
## 2345.526
```

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

```
##     var1
## 1426.611
```

Problem 3 IMH for simple logistic regression problem

```
# 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
```

```r
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)
}
```

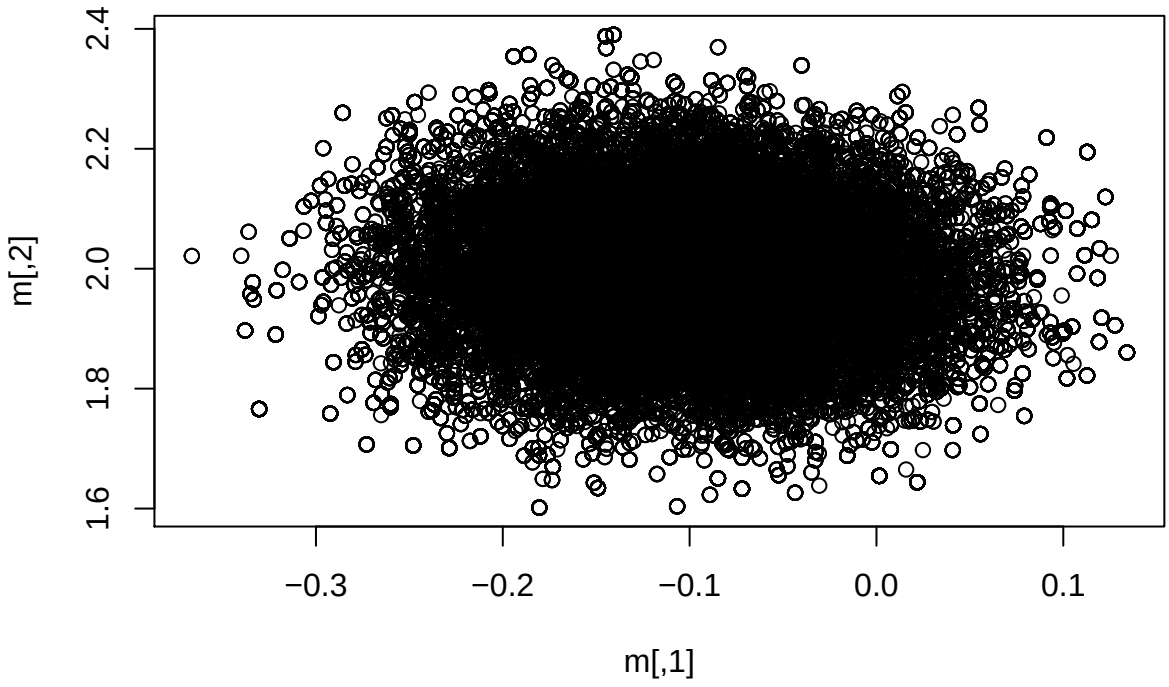
In this task, we will use independent metropolis-hastings sampling for the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{y})$ using a $N(\hat{\theta}, \delta\hat{\Sigma})$ proposal distribution.

In R:

```r
# Independent sampler
IndepMH <- function(prob, 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 <- prob(theta)/dmvnorm(theta, mean=theta ,sigma=sigma)
  # 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 <- prob(new)/dmvnorm(new, mean=theta, sigma=sigma)
    frac <- min(1.0, 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, 30000, 0.6)
```

```
## [1] "accept rate : 0.584266666666667"
```

```
plot(m)
```



Calculate ESS

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

```
##    var1
## 1887.35
```

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

```
##     var1
## 1860.481
```

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.

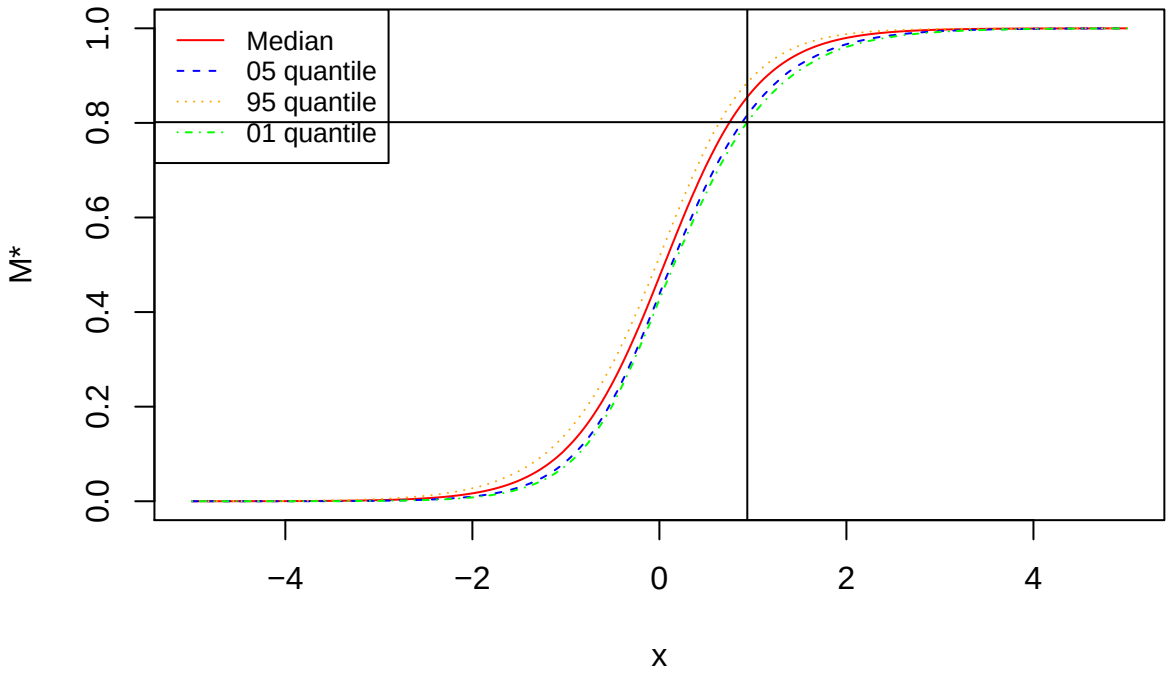
We use the following algorithm to solve this problem:

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)
```




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

```
## [1] 0.94
```

Bibliography