# Module 9: Solutions

## Exercise 1: Minimise auto-correlation

Write a function in R that uses a Metropolis-Hastings algorithm to simulate from a standard normal distribution, using normally distributed proposals centred at the current value (in other words, this is a random walk Metropolis algorithm). Make sure your code allows you to adjust the standard deviation of the proposal distribution and monitor the acceptance probability. (**Hint:** You already implemented this in the exercises for module 7 (see the solution if you didn't finish the exercise) and you may use this as the main part of your function). The skeleton of such a function is given below:

```
library(coda)
myMH <- function(N, sigma){
  x <- rep(0, N) # Empty Markov chain
  a <- 0 # Number of accepted proposals
  # Below implement the M-H alg. and add 1 to `a` every time you accept
  x[1] <- 0 ## initial value
  for(i in 2:N){
    y <- rnorm(1, x[i-1], sigma)
    u <- runif(1)
    H <- dnorm(y)/dnorm(x[i-1])
    if(u<H){
      x[i] <- y
      a <- a+1
    }else{
      x[i] <- x[i-1]
    }
  }
  # Finally return a list of the results
  return(list(x=mcmc(x), a=a/(N-1)))
}
# A short run with proposal std. dev. 1
rslt <- myMH(10000, 1)
```

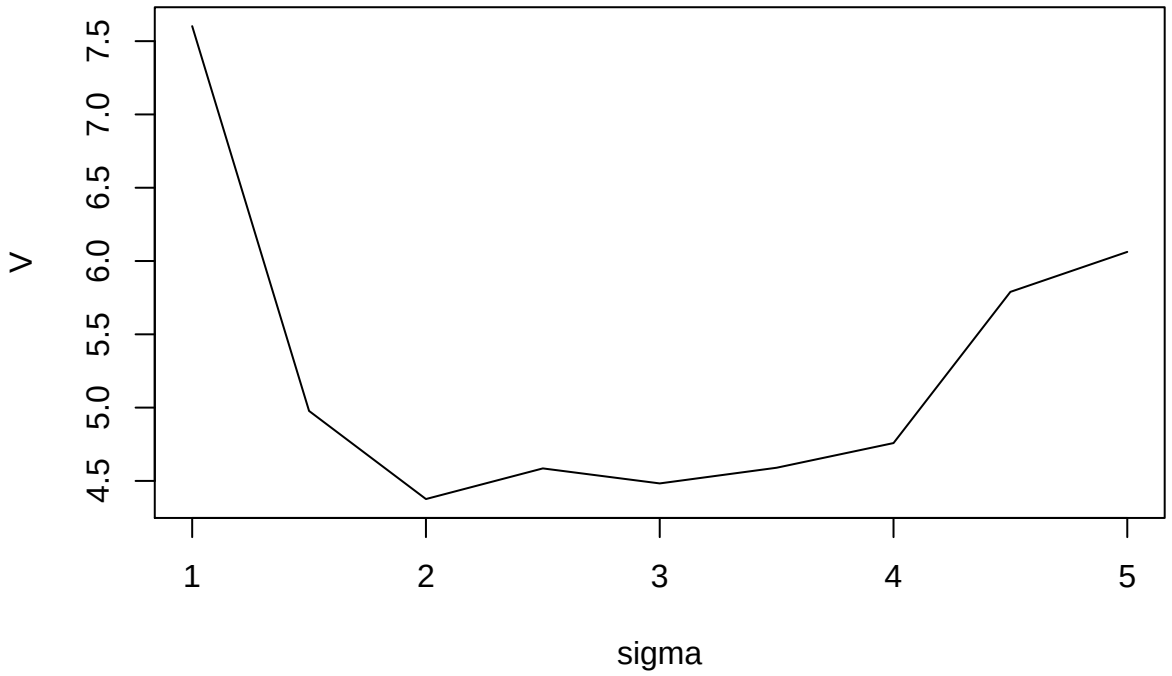
To plot the auto-correlation function use the function `acf()` in R. To approximately calculate $V = 1 + 2\sum_{m=1}^{\infty} \rho_m$ use the following command in R:

```
V <- 2 * sum(acf(rslt$x, plot = FALSE)$acf) - 1
```

Make some experiments and find the standard deviation for the proposal distribution that gives the smallest value of $V$. What is the acceptance probability in this case? (Make sure that you have run the MH-algorithm long enough (really long), so that the auto-correlation function is well estimated.)

```
sigma <- seq(1, 5, by = .5)
V <- rep(0, length(sigma))
a <- rep(0, length(sigma))
for(i in 1:length(sigma)){
  rslt <- myMH(100000, sigma[i])
  x <- rslt$x
  acf_val <- acf(x, plot = FALSE)$acf
  V[i] <- 2 * sum(acf_val) - 1
```

```
  a[i] <- rslt$a
}
plot(sigma, V, type = "l")
```




```
a[which.min(V)]
```

```
## [1] 0.498895
```

## Exercise 2: Beetles

This example is concerned with different beetles exposed in 5 hours to different concentrations of carbon disulphide. The data contains for 481 beetles, in the first column, the concentration, and in the second column, the status of the beetles after the exposure. The status is coded as zeros (dead beetles) and ones (survived beetles). Read in the data set `beetles.dat` from the course website:

```
data_url <- "https://asta.math.aau.dk/course/bayes/2018/?file=beetles.dat"
beetles <- read.table(data_url)
```

Let $x_i > 0$ and $y_i \in \{0, 1\}$ denote the concentration and status for the $i$th beetle, respectively. We use a logistic regression model for the data of survived and dead beetles, i.e., we condition on the concentrations and assume that the probability that the $i$th beetle survives its given dose $x_i$ is

$$\frac{\exp(a + bx_i)}{1 + \exp(a + bx_i)}$$

where $a$ and $b$ are real parameters. In R you can define the data vecors and log-likelihood function as:

```
beet_x <- beetles[,1]
beet_y <- beetles[,2]
llik = function(a, b){
```

```
  ## BEWARE: We refer to global variables beet_x and beet_y here!
  return(sum(beet_y*(a+b*beet_x)) - sum(log(1+exp(a+b*beet_x))))
}
```

Now, let Cauchy($\gamma$) denote the symmetric Cauchy distribution with scale parameter $\gamma > 0$; this distribution has density

$$\pi(c|\gamma) = \frac{1}{\pi}\frac{\gamma}{\gamma^2 + c^2}, \qquad c \in \mathbf{R}.$$

Assume a priori that $a \sim$ Cauchy(10) and $b \sim$ Cauchy(2.5) are independent (the argument for using this prior is given in a 2008 paper by Andrew Gelman and coauthors in Annals of Applied Statistics). Then construct a Metropolis-Hastings algorithm which generates a sample from the posterior distribution of $(a, b)$.

**Hint:** You have to work with the **logarithm** of the Hastings ratio for numerical stability. The logarithm of the prior density can be calculated as

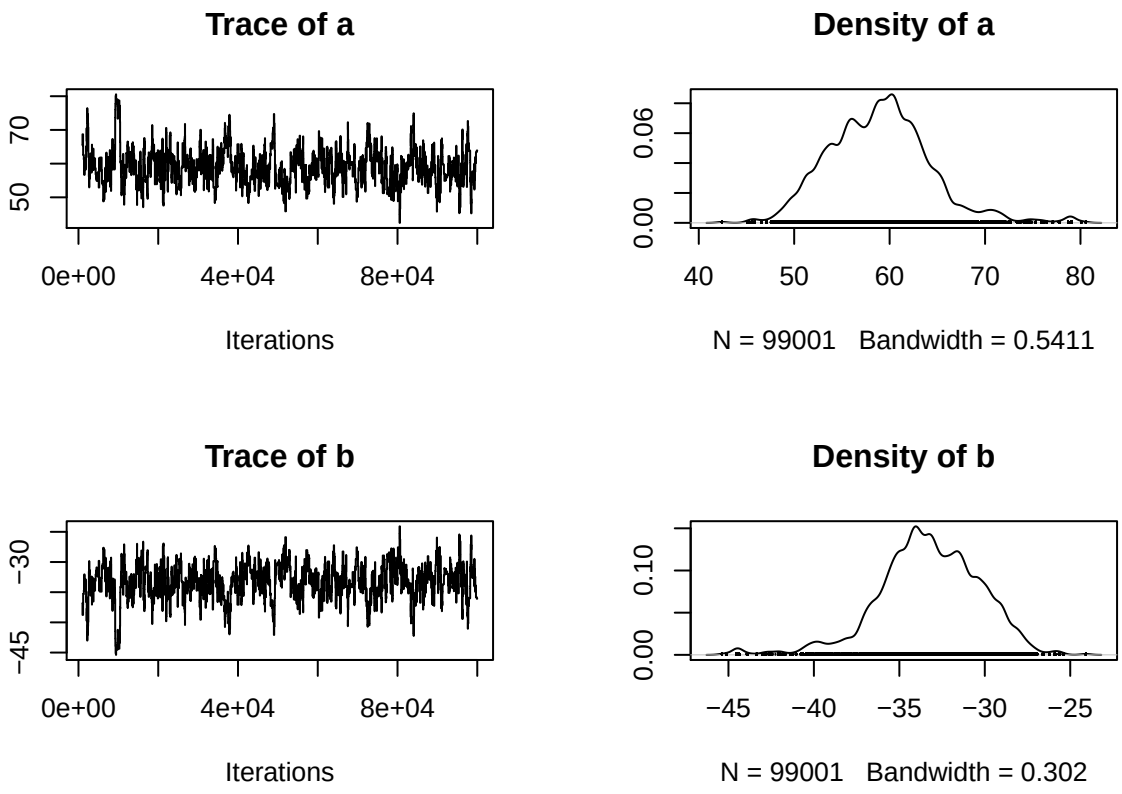
```
lprior <- function(a, b){
  dcauchy(a, scale = 10, log = TRUE) + dcauchy(b, scale = 2.5, log = TRUE)
}
```

```
lposterior <- function(a, b){
  llik(a, b) + lprior(a, b)
}
myMH2 <- function(N, sigma = c(1,1), a0=1, b0=1){
  a <- b <- rep(0, N) # Empty Markov chain
  a[1] <- a0 ## initial value
  b[1] <- b0 ## initial value
  for(i in 2:N){
    a_new <- rnorm(1, a[i-1], sigma[1])
    b_new <- rnorm(1, b[i-1], sigma[2])
    logH <- lposterior(a_new, b_new) - lposterior(a[i-1], b[i-1])
    if(log(runif(1))<logH){
      a[i] <- a_new
      b[i] <- b_new
    }else{
      a[i] <- a[i-1]
      b[i] <- b[i-1]
    }
  }
  # Finally return the results
  return(mcmc(cbind(a,b)))
}
# Try it out:
chain <- myMH2(100000, sigma = c(4,4), a0 = 60, b0 = -30)
chain <- window(chain, start = 1000)
summary(chain)
```

```
##
## Iterations = 1000:1e+05
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 99001
##
## 1. Empirical mean and standard deviation for each variable,
```

```
##    plus standard error of the mean:
##
##     Mean    SD Naive SE Time-series SE
## a  58.96 5.342 0.016978         0.4255
## b -33.27 2.998 0.009529         0.2383
##
## 2. Quantiles for each variable:
##
##     2.5%    25%    50%    75%  97.5%
## a  49.64  55.30  58.89  62.13  70.98
## b -40.03 -35.03 -33.23 -31.22 -28.04
```

```
plot(chain)
```




## Exercise 3: The banana

Construct a Metropolis-Hastings algorithm which has the following (unnormalised) invariant two-dimensional density

$$\pi(x, y) \propto \exp(-x^2/200 - (y + 0.1 * x^2 - 10)^2) \qquad x, y \in \mathbf{R}.$$

Notice that $\pi(x, y)$ is symmetric in $x$. Do your simulations reproduce this symmetry?

```
lf <- function(x, y){
  -x^2/200 - (y + 0.1*x^2 - 10)^2
}
myMH3 <- function(N, sigma = c(1,1), x0=0, y0=0){
  x <- y <- rep(0, N) # Empty Markov chain
```

```
  x[1] <- x0 ## initial value
  y[1] <- y0 ## initial value
  for(i in 2:N){
    x_new <- rnorm(1, x[i-1], sigma[1])
    y_new <- rnorm(1, y[i-1], sigma[2])
    logH <- lf(x_new, y_new) - lf(x[i-1], y[i-1])
    if(log(runif(1))<logH){
      x[i] <- x_new
      y[i] <- y_new
    }else{
      x[i] <- x[i-1]
      y[i] <- y[i-1]
    }
  }
  # Finally return a list of the results
  return(cbind(x,y))
}
# Try it out:
library(coda)
chain <- myMH3(10000, sigma = c(2,2), x0 = 0, y0 = 0)
plot(chain[,1], chain[,2], ylim = c(min(chain[,2]), 20), col = rgb(0,0,0,.1))
xx <- seq(-20, 20, by=.5)
yy <- seq(-10, 20, by=.5)
contour(xx, yy, outer(xx, yy, lf), levels = seq(-50, 0, by = 10), add = TRUE, col = 2)
```