# Portfolio 8: Metropolis-Hastings Algorithm

Kieran Morris

For this portfolio, we will implement the Metropolis-Hastings algorithm to sample from the posterior distribution of the parameters of a logistic regression model, our toy dataset for today is the Pima Indians Diabetes dataset from the `mlbench` package.

```
library(mlbench)
data("PimaIndiansDiabetes")
head(PimaIndiansDiabetes)
```

```
##   pregnant glucose pressure triceps insulin mass pedigree age diabetes
## 1        6     148       72      35       0 33.6    0.627  50      pos
## 2        1      85       66      29       0 26.6    0.351  31      neg
## 3        8     183       64       0       0 23.3    0.672  32      pos
## 4        1      89       66      23      94 28.1    0.167  21      neg
## 5        0     137       40      35     168 43.1    2.288  33      pos
## 6        5     116       74       0       0 25.6    0.201  30      neg
```

As you can see the dataset has 8 features and a binary response variable `diabetes`, which takes values `pos` and `neg`. Which we will probably change later on! We setup a logistic regression model:

## Task 1 - Metropolis Hastings

$$Pr_{\alpha,\beta}(Y_i = 1) = \frac{1}{1 + e^{-\alpha - \beta^T x_i}}$$

with the following likelihood function:

$$L_n(\alpha, \beta) = \prod_{i=1}^{n} Pr_{\alpha,\beta}(Y_i = y_i)$$

and posterior distribution:

$$\pi(\alpha, \beta | y) \propto L_n(\alpha, \beta)\pi(\alpha, \beta)$$

the log posterior given by

$$\log \pi(\alpha, \beta | y) = \sum_{i=1}^{n} \left( y_i \log \frac{1}{1 + e^{-\alpha - \beta^T x_i}} + (1 - y_i) \log \frac{e^{-\alpha - \beta^T x_i}}{1 + e^{-\alpha - \beta^T x_i}} \right) + \pi(\alpha, \beta)$$

where $\pi(\alpha, \beta)$ is the prior.

## 1. Choice of Distribution

We will use the suggested choice of proposal distrubution, a multivariate normal distribution at each $z$:

$$Q(z, dz') = \mathcal{N}_{p+1}(z, c\mathbf{\Sigma}_n)$$

for tuning parameter $c > 0$ and

$$\mu_n = \underset{(\alpha,\beta)\in\mathbb{R}^p}{\arg\max} \log \pi(\alpha, \beta|y), \qquad \mathbf{\Sigma}_n = -(\mathbf{H}_n(\mu_q))^{-1}$$

where $\Sigma_n$ is the hessian of the log posterior at $\mu_q$ and can be estimated by the covariance matrix of the coefficients under $\pi(\alpha, \beta|y_0)$.

To get the covariance matrix, we first fit a logistic regression model to the data and output it:

```
data <- PimaIndiansDiabetes
y <- as.numeric((data$diabetes))-1
X <- scale(data[,1:8])
fit <- glm(y ~ X, family = "binomial")
z_0 <- fit$coefficients
Sigma_n <- summary(fit)$cov.scaled
```

## 2. Implementing Metropolis Hastings

We attempt to manually perform the Metropolis-Hastings algorithm. So we begin by coding up a function which produces the posterior likelihood, with a flat prior:

```
library(mvtnorm)
posterior_likelihood <- function(par, X, y){
  alpha <- par[1]
  beta <- par[2:9]
  p <- 1 - 1 / (1 + exp(alpha + beta%*%t(X)))
  likelihood <- exp(sum(dbinom(y, size=1, prob=p, log=TRUE)) )
  return(likelihood)
}
```

Since we now have a proposal distribution and a posterior likelihood, we can now run the MH algorithm:

```
metropolis_hastings <- function(z_0,c, sigma_n, tmax, X, y){
  zs <- matrix(NA, nrow =tmax, ncol = length(z_0))
  z_current <- z_0
  z_current_ll <- posterior_likelihood(z_current, X,y)
  sigma <- c * sigma_n
  accepted <- 0
  for (i in 1:tmax){
    z_new <- rmvnorm(1, z_current, sigma)
    z_new_ll <- posterior_likelihood(z_new, X, y)
    alpha <- log(z_new_ll)- log(z_current_ll)
    if (runif(1) < min(1,exp(alpha))){

      z_current <- z_new
      z_current_ll <- z_new_ll
      accepted <- accepted + 1
```

```
    }
    zs[i,] <- z_current
  }
  return(list(zs = zs, acceptance_rate = accepted/tmax))
}
```

we begin with $c = 1$ and run the algorithm for 10000 iterations:

```
mh <- metropolis_hastings(z_0, 1, Sigma_n, 10000, X, y)

mh$acceptance_rate
```

```
## [1] 0.1794
```

obviously this isn't close to our ideal 0.234 acceptance rate, so we will need to try a range of $c$ values to get a better acceptance rate:

```
c_values <- seq(0.1, 2, 0.1)
acceptance_rates <- numeric(length(c_values))
for (i in 1:length(c_values)){
  acceptance_rates[i] <- metropolis_hastings(z_0, c_values[i], Sigma_n, 10000, X, y)$acceptance_rate
}
#Output the c values and their acceptance error
cbind(c_values, acceptance_rates)
```

```
##       c_values acceptance_rates
##  [1,]      0.1           0.6521
##  [2,]      0.2           0.5250
##  [3,]      0.3           0.4426
##  [4,]      0.4           0.3780
##  [5,]      0.5           0.3231
##  [6,]      0.6           0.2754
##  [7,]      0.7           0.2355
##  [8,]      0.8           0.2088
##  [9,]      0.9           0.1906
## [10,]      1.0           0.1757
## [11,]      1.1           0.1556
## [12,]      1.2           0.1366
## [13,]      1.3           0.1305
## [14,]      1.4           0.1112
## [15,]      1.5           0.1045
## [16,]      1.6           0.0986
## [17,]      1.7           0.0805
## [18,]      1.8           0.0795
## [19,]      1.9           0.0754
## [20,]      2.0           0.0627
```

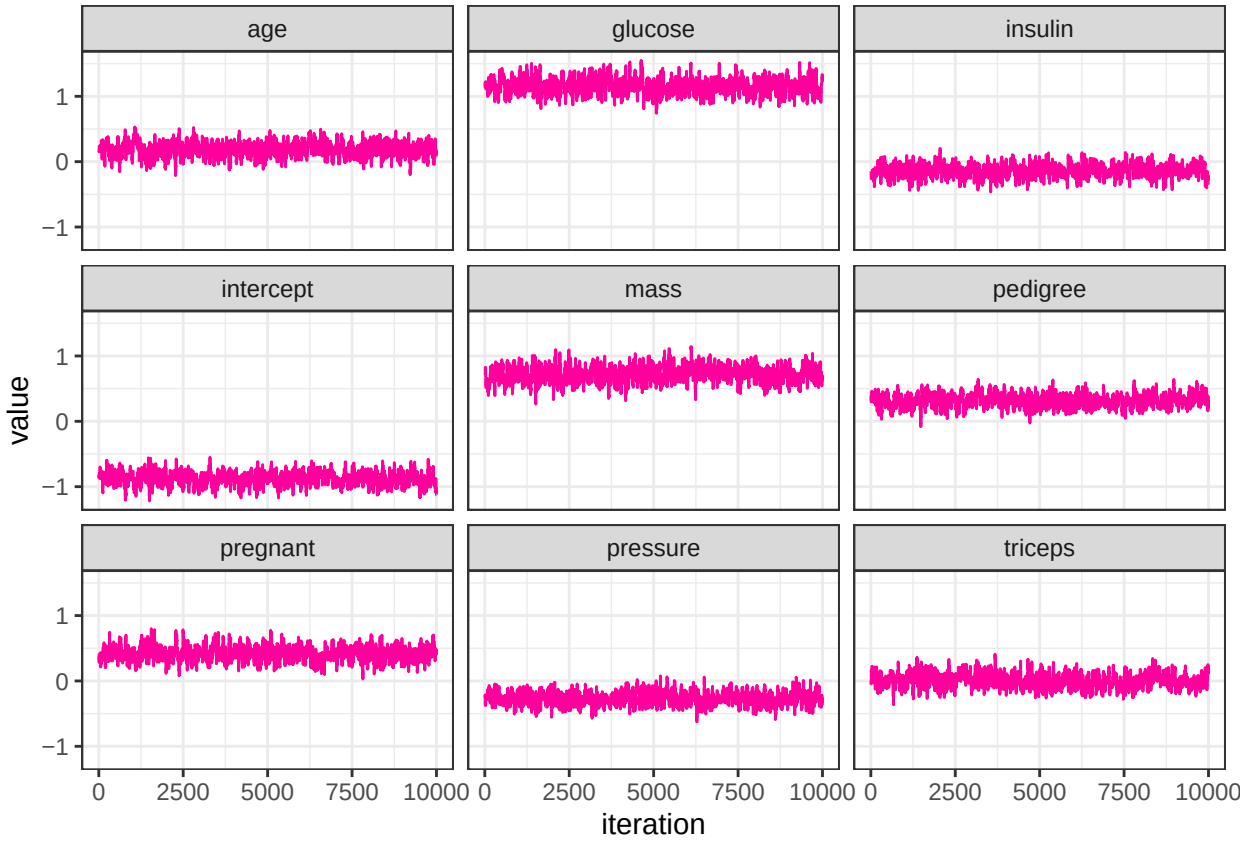
Our best choice for $c$ is then 0.8. We can then run the Metropolis-Hastings algorithm with this value of $c$:

```
mh <- metropolis_hastings(z_0, 0.8, Sigma_n, 10000, X, y)
```

## 3. Evaluating convergence with the optimal c

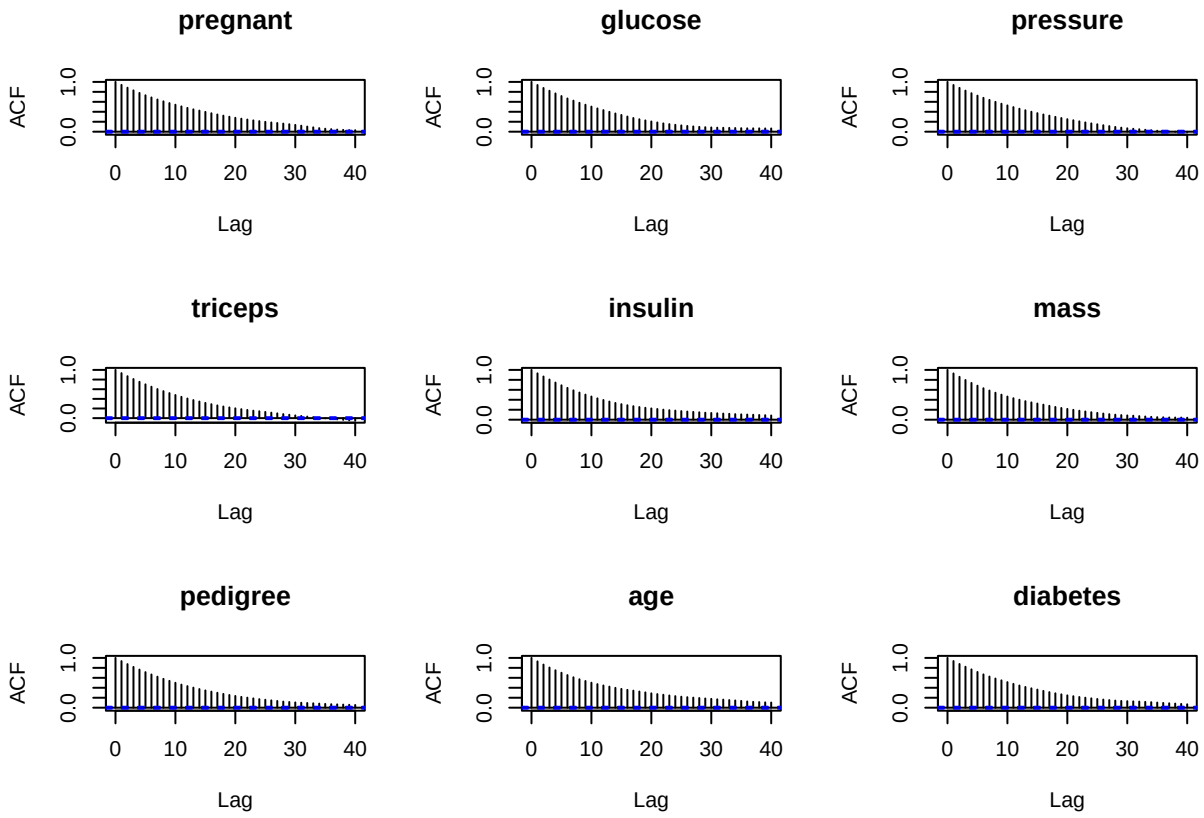
Let's check the convergence of our algorithm, we begin by looking at the trace plots and autocorrelation plots:

```
library(dplyr)
library(tidyr)
library(ggplot2)

colnames(mh$zs) <- c("intercept", colnames(data)[1:8])
zs_plot <- mh$zs %>%
  as_tibble() %>%
  mutate(iteration = 1:10000) %>%
  pivot_longer(cols = 1:9)

ggplot(zs_plot, aes(iteration, value)) +
  geom_line(colour = "#ff009d") +
  facet_wrap(vars(name))
```




```
par(mfrow = c(3,3))
for (i in 1:9){
```

```
  acf(mh$zs[,i], main = colnames(data)[i])
}
```



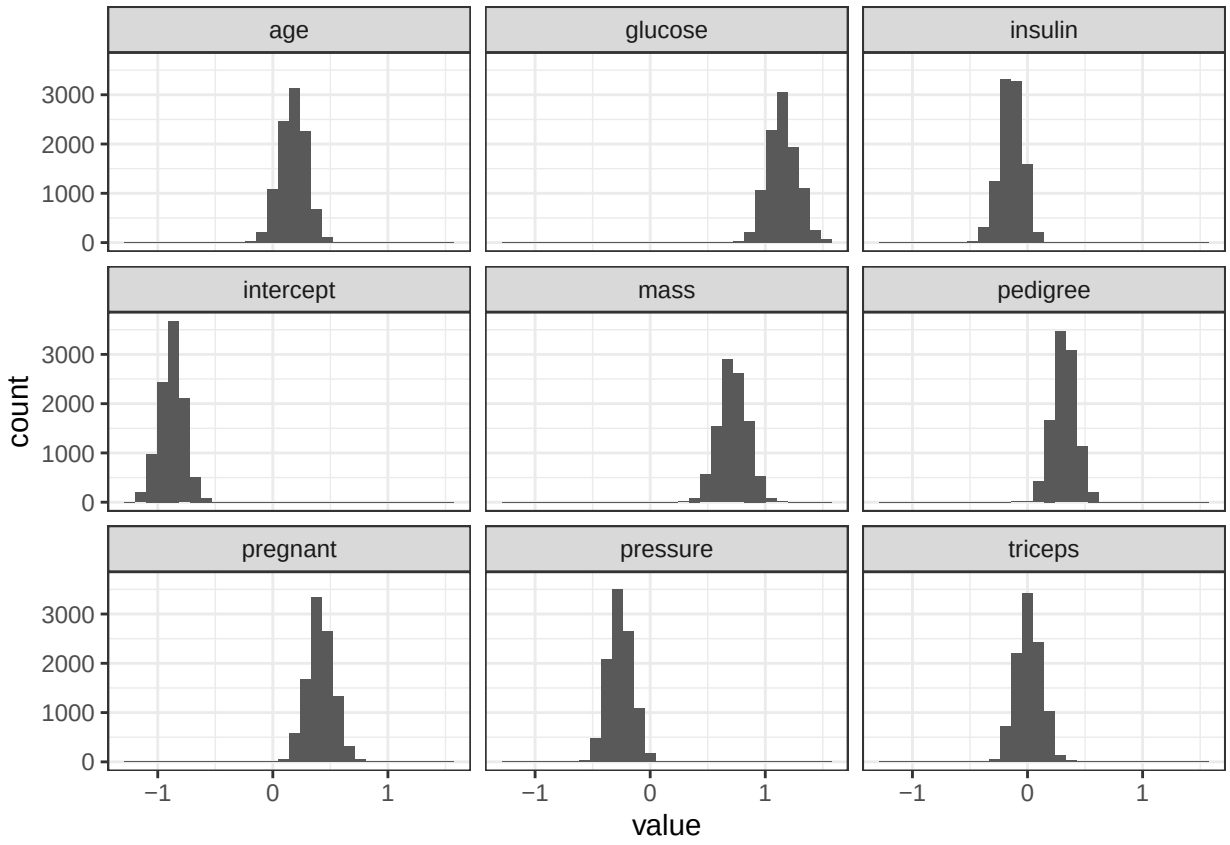

Since we did the work finding an optimal $c$ value prior to modelling, we have a pretty strong acceptance rate and as we can see from the plots we have a tight distribution. The autocorrelation plots also are good indicators as they trail off pretty quickly.

## 5. Marginal posterior distributions

Now that we're happy with our model, we can plot the marginal posterior distributions for each parameter:

```
ggplot(zs_plot, aes(value)) +
  geom_histogram() +
  facet_wrap(vars(name))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




# Task 2 - Multiple Proposal Distributions

We now change our model to allow for mutliple proposal distributions, we will use a mixture of normal distributions:

$$Q(z, dz') = \sum_{i=1}^{k} w_i \mathcal{N}_{p+1}(z, c_i \mathbf{\Sigma}_n)$$

where $w_i$ are the weights of the mixture and $c_i$ are the tuning parameters. We will use $k = 3$ and $w_i = 1/3$ for all $i$.

## 1. Implementing the MH algorithm

We first need to build a function to calculate the mixture of normal distributions:

```
mixture_normal <- function(z, c, sigma_n){
  k <- length(c)
  zs <- matrix(NA, nrow = k, ncol = length(z))
  for (i in 1:k){
    zs[i,] <- rmvnorm(1, z, c[i] * sigma_n)
  }
  return(zs)
}
```