portfolio_8

2023-04-26

Metropolis-Hastings algorithm, Gibbs sampler and Metropolis-within-Gibbs

Let's load in the PimaIndiansDiabetes dataset which we are going to use in this portfolio:

```
library(mlbench)
data(PimaIndiansDiabetes)
PID <- PimaIndiansDiabetes

PID["diabetes"] <- ifelse(PID[["diabetes"]] == "pos", 1, 0)
head(PID)
```

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

```
X <- as.matrix(PID[, -9])
y <- PID[, 9]
```

Let's first define the prior distribution:

```
prior_mu <- rep(0, 9)    # prior mean
prior_sigma <- diag(9)    # prior covariance
```

For our proposal distribution Q we will use the proposal distribution mentioned in the worksheet:

```
library(MASS)
library(mvtnorm)
library(numDeriv)

# The log-likelihood function
log_likelihood <- function(params, X, y) {
  z <- params[1] + X %*% params[2:9]
  sum(y*z - log(1 + exp(z)))
}

# The log posterior distribution
log_posterior <- function(params) {
  z <- params[1] + X %*% params[2:9]
  ll <- sum(y*z - log(1 + exp(z)))
  log_prior <- dmvnorm(params, mean = prior_mu, sigma =prior_sigma, log = TRUE)
  log_post <- ll + log_prior
  return(log_post)
```

```r
}

# Negative version of the log posterior (for when using optim)
n_log_posterior <- function(params) {
  z <- params[1] + X %*% params[2:9]
  ll <- sum(y*z - log(1 + exp(z)))
  log_prior <- dmvnorm(params, mean = prior_mu, sigma =prior_sigma, log = TRUE)
  log_post <- ll + log_prior
  return(-log_post)
}

# For calculating the proposal ratio
proposal_ratio <- function(current, proposal, proposal_cov){
    dmvnorm(current, mean = proposal, sigma = proposal_cov, log = TRUE) - dmvnorm(proposal, mean = curr
}

# We set the number of iterations and burn-in period
num_iterations <- 10000
burn <- 1000

# We set up the proposal distribution
mu_n <- optim(rep(0.1,9), n_log_posterior)$par
sigma_n <- -solve(hessian(func=log_posterior, x=mu_n))

# Initial parameters
params <- rep(0, 9)
chain <- matrix(0, nrow=num_iterations, ncol=9)
chain[1,] <- params
accept <- 0

# The Metropolis-Hastings algorithm
for (i in 2:num_iterations) {
  # We sample a proposal from the proposal distribution
  proposal <- mvrnorm(1, mu_n, sigma_n)

  # We compute the acceptance ratio
  log_ratio <- log_likelihood(proposal, X, y) - log_likelihood(params, X, y) + dmvnorm(proposal, mean =

  # We accept or reject the proposal
  if (log(runif(1)) < exp(log_ratio)) {
    params <- proposal
    accept <- accept + 1
  }

  # We store current parameter value in the chain
  chain[i,] <- params
}
# We discard the burn in phase
chain <- chain[-c(1:burn),]

# We print the acceptance rate
cat("Acceptance rate:", accept / num_iterations, "\n")
```

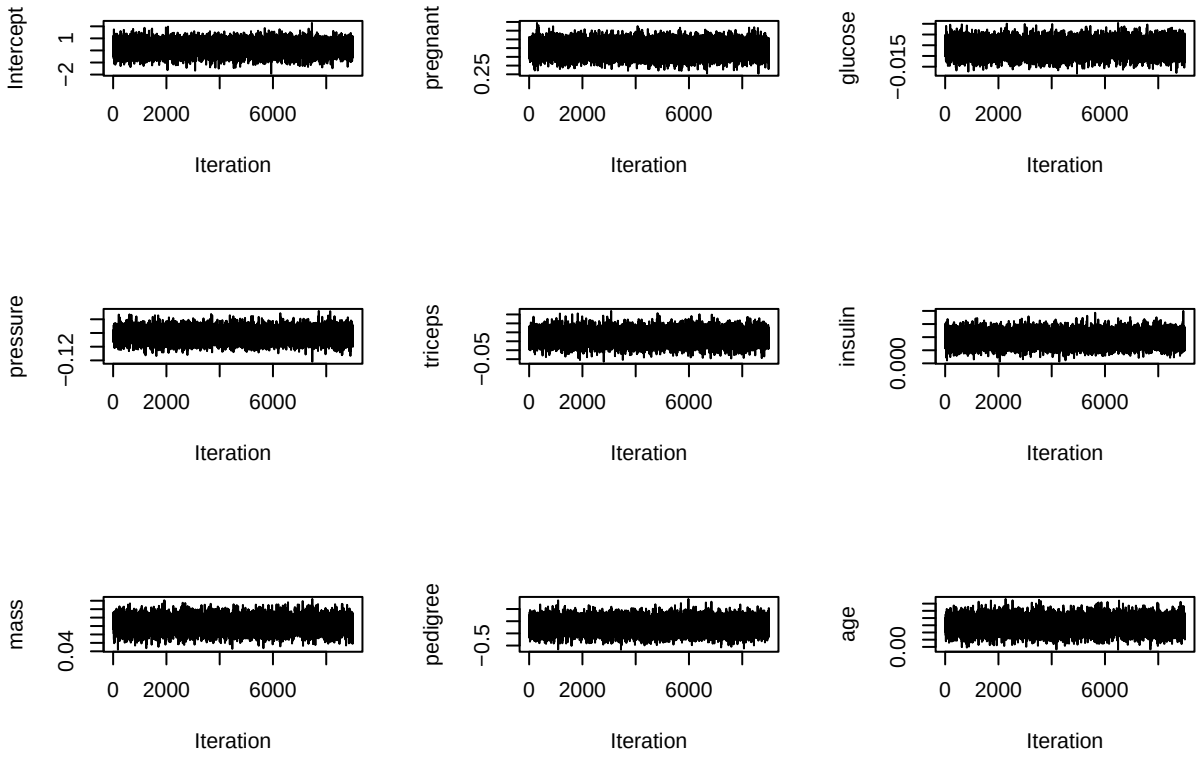
```
## Acceptance rate: 0.9999
```

```r
# We print some summary statistics
post_mean <- apply(chain, 2, mean)
post_sd <- apply(chain, 2, sd)
post_quantiles <- apply(chain, 2, quantile, c(0.025, 0.5, 0.975))
post_summary <- rbind(post_mean, post_sd, post_quantiles)
colnames(post_summary) <- c( "Intercept" , colnames(PID[, -9]) )
rownames(post_summary) <- c("Mean", "SD", "2.5%", "50%", "97.5%")
print(post_summary)
```

```
##        Intercept   pregnant       glucose     pressure      triceps
## Mean   0.1831758 0.39478149 -0.0063938845 -0.082998315 -0.025349682
## SD     0.5061126 0.03937814  0.0032751503  0.008843495  0.007413018
## 2.5%  -0.8094290 0.31727848 -0.0127712199 -0.100673626 -0.039584944
## 50%    0.1827937 0.39498825 -0.0063946332 -0.082889939 -0.025357332
## 97.5%  1.1776416 0.47200285  0.0001642306 -0.065810461 -0.010957062
##            insulin       mass   pedigree         age
## Mean  0.0037889050 0.10420374  0.3133310 0.030495306
## SD    0.0009965893 0.01634966  0.2758602 0.009830527
## 2.5%  0.0018519143 0.07228889 -0.2353902 0.011558894
## 50%   0.0037769153 0.10400536  0.3127754 0.030380534
## 97.5% 0.0057140922 0.13578985  0.8570115 0.049790451
```

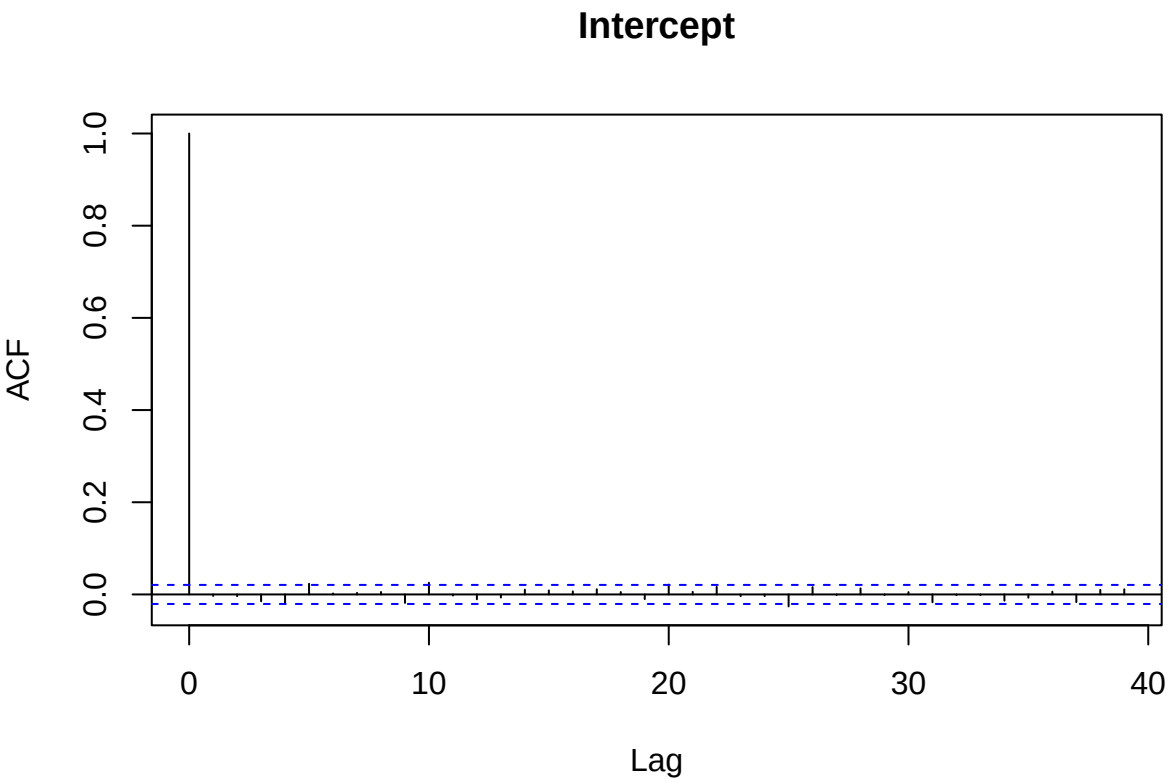
Above we can see some summary statistics on chain for the parameters aswell as the acceptance rate. We note the sd of the chains for the parameters are quite small. We also note that the acceptance rate is extremely high could mean that our proposal distribution is well tuned. We will now investigate further whether our MCMC chain has properly converged. Let's now produce some trace plots of the parameters:

```r
par(mfrow=c(3,3))
plot(chain[,1], type="l", xlab="Iteration", ylab="Intercept")
plot(chain[,2], type="l", xlab="Iteration", ylab="pregnant")
plot(chain[,3], type="l", xlab="Iteration", ylab="glucose")
plot(chain[,4], type="l", xlab="Iteration", ylab="pressure")
plot(chain[,5], type="l", xlab="Iteration", ylab="triceps")
plot(chain[,6], type="l", xlab="Iteration", ylab="insulin")
plot(chain[,7], type="l", xlab="Iteration", ylab="mass")
plot(chain[,8], type="l", xlab="Iteration", ylab="pedigree")
plot(chain[,9], type="l", xlab="Iteration", ylab="age")
```

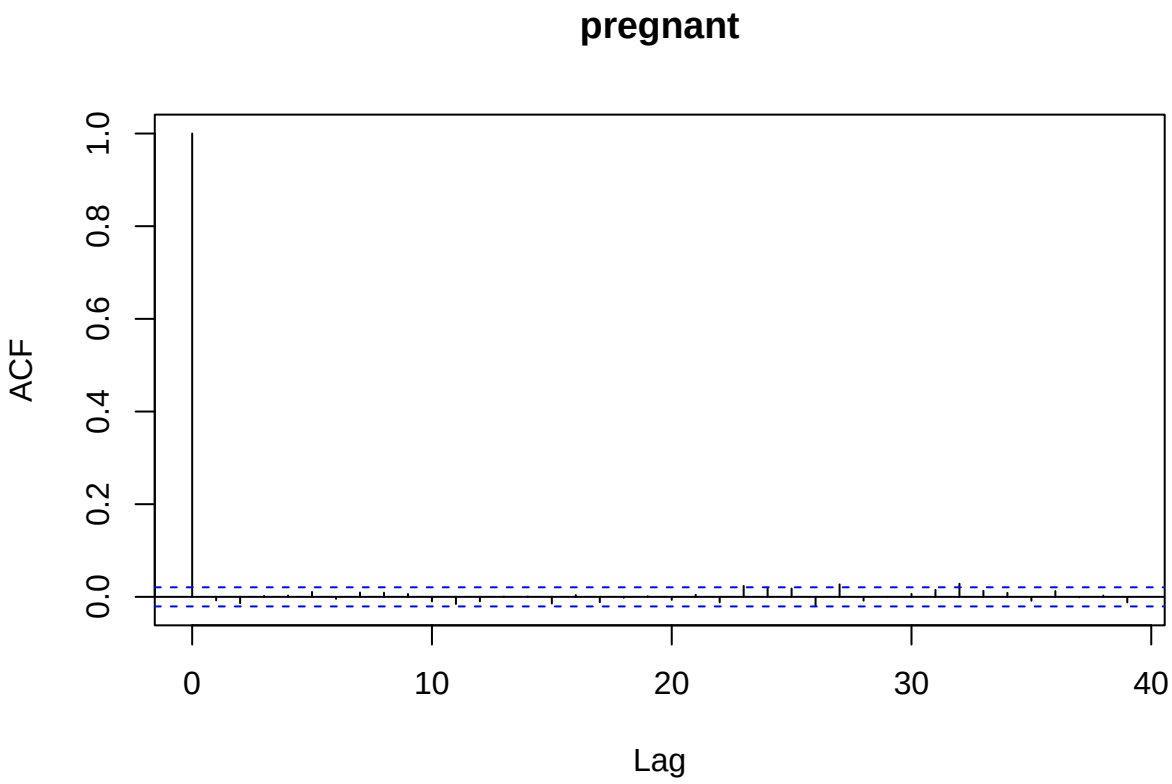


These trace plots look great as they are stationary, look very random and there are no obvious trends in trace. This is indicative of a MCMC chain that has properly converged. Let's now look at the acf plots for all the parameters:

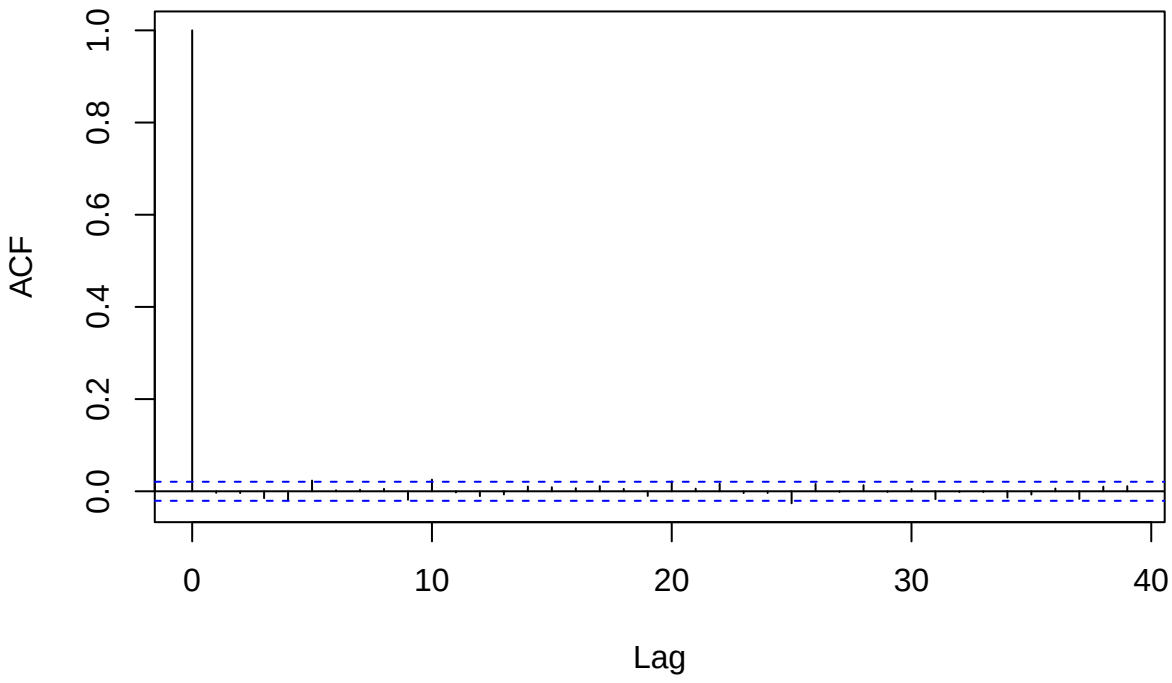
```
acf(chain[,1], main="Intercept")
```



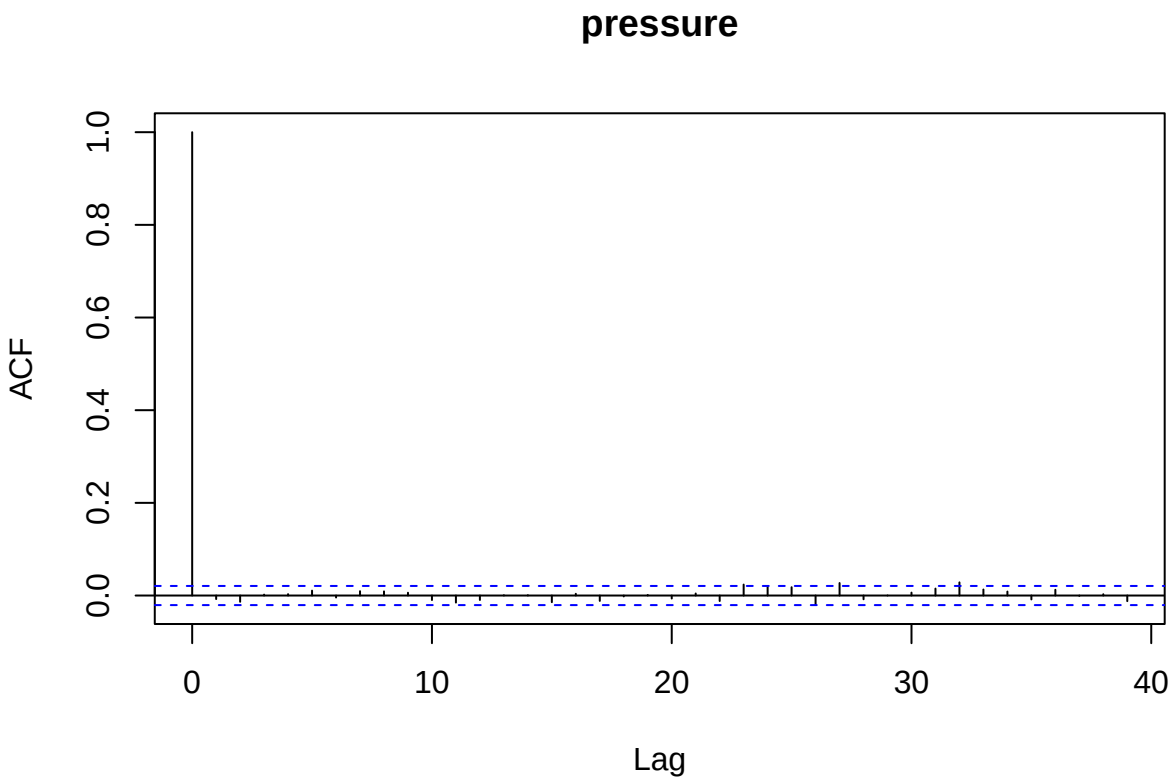
```
acf(chain[,2], main="pregnant")
```



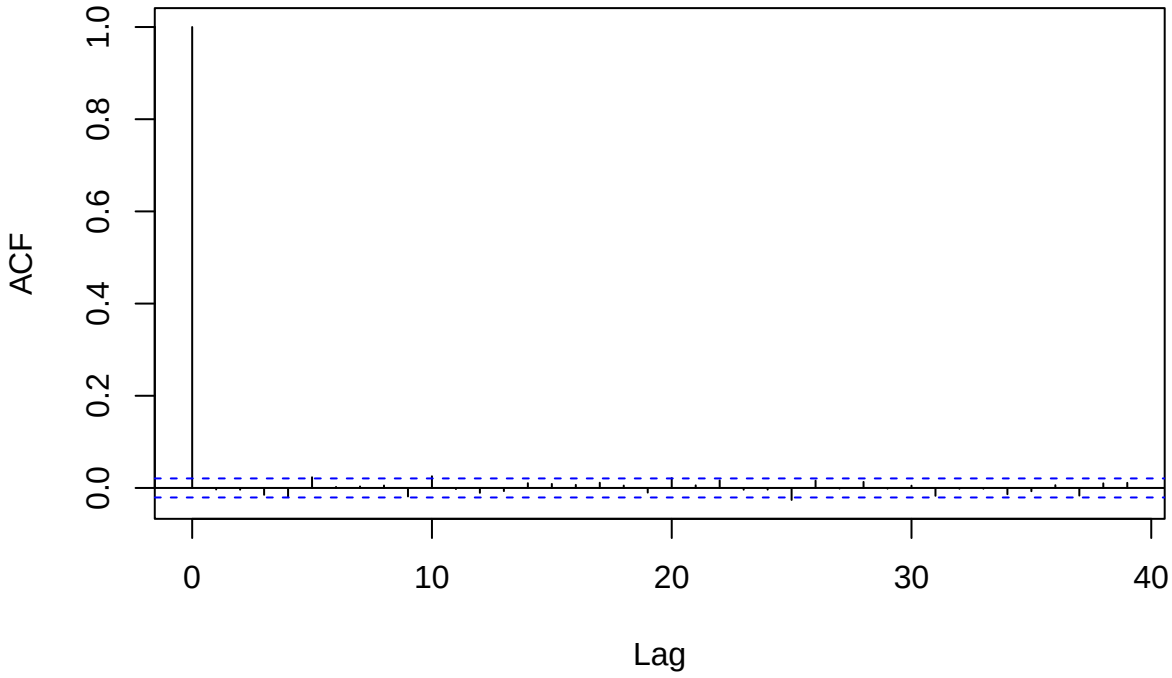

```
acf(chain[,1], main="glucose")
```



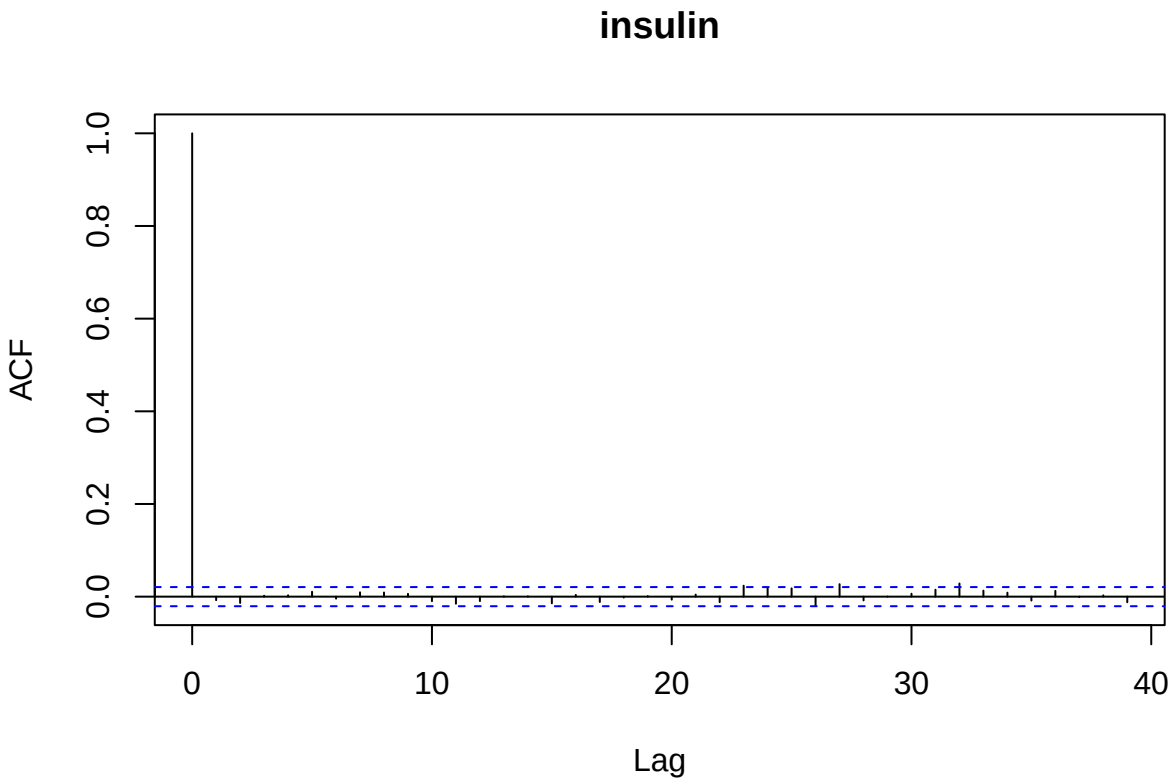
```
acf(chain[,2], main="pressure")
```



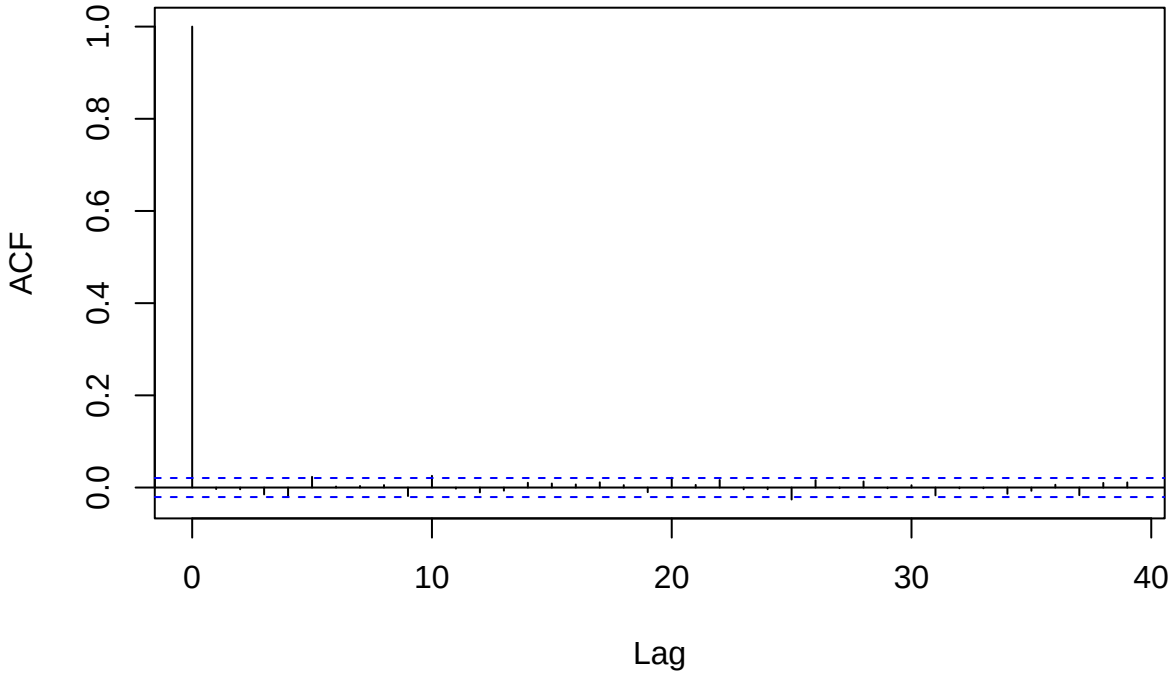

```
acf(chain[,1], main="triceps")
```



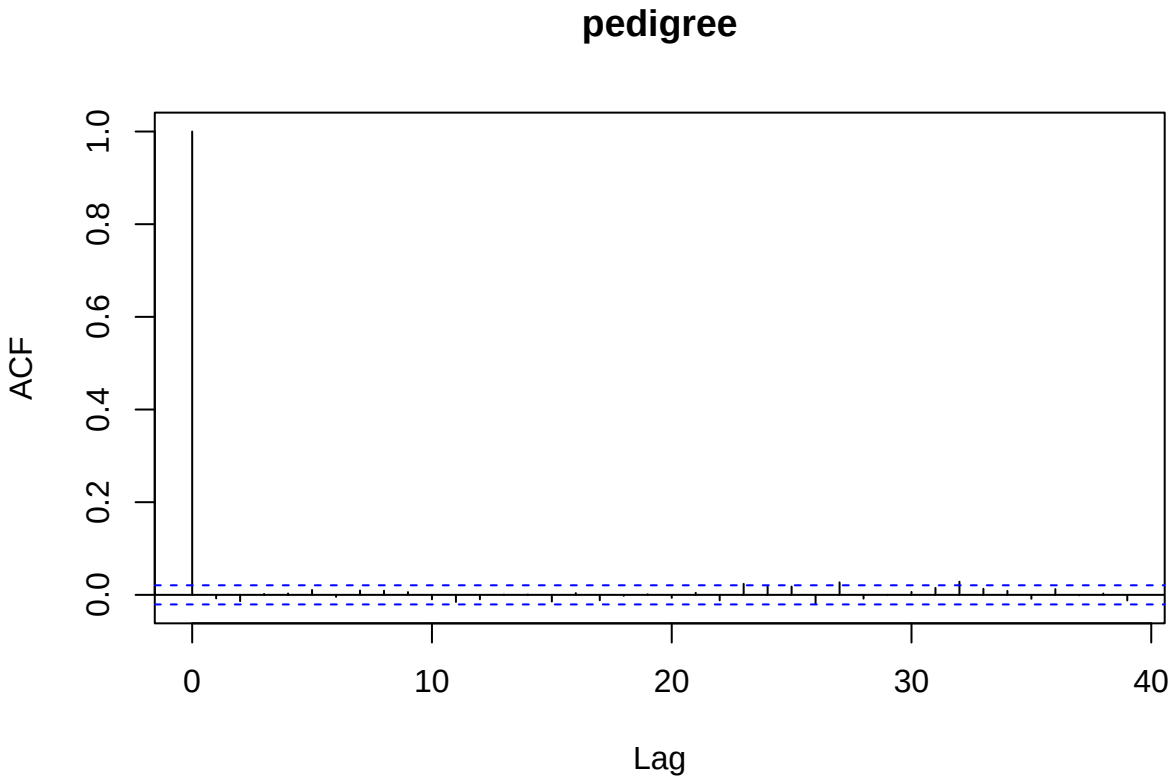
```
acf(chain[,2], main="insulin")
```



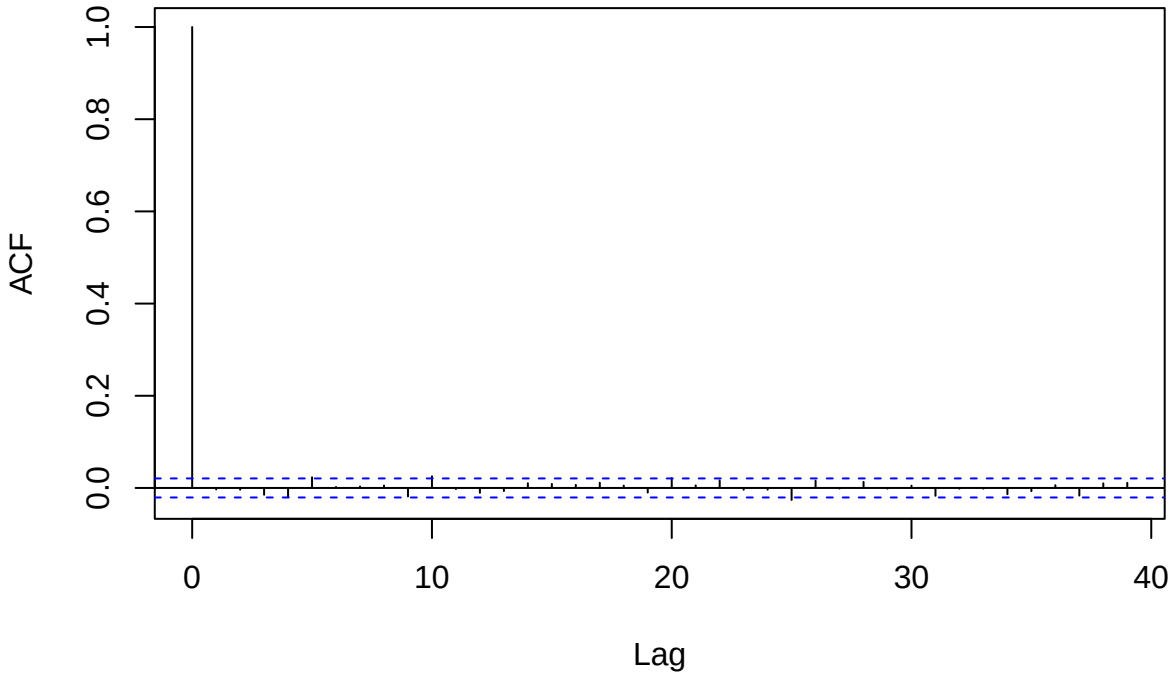

```
acf(chain[,1], main="mass")
```



```
acf(chain[,2], main="pedigree")
```




```
acf(chain[,1], main="age")
```



The acf plots also look great with the ACF dropping off instantly for all parameters, ie. the chain shows very low

autocorrelation. This also indicates that the MCMC chain has properly converged. Finally, let's plot the estimated marginal posterior distribution for each of the parameters:

```
names <- c( "Intercept" , colnames(PID[, -9]) )
# Plot the estimated marginal posterior distribution for each parameter
par(mfrow=c(3,3))
for (i in 1:9) {
  density <- density(chain[, i])
  plot(density, main=names[i], xlab="value", ylab="density", ylim=c(0, max(density$y)*1.1))
}
```