MATH 640: Exam 2

Name:_____

Instructions

1. Do not discuss this midterm with anyone other than Professor Meyer or the TA; and you may only ask clarifying questions.
2. Your responses to this exam *must* be typed.
3. This cover sheet *must* be the first page of your submission.
4. If you wish to cite a result or derivation from lecture, you may do so but be sure it is clearly cited (list the slide from the Note Set or the example) and relevant.
5. Please submit the exam to the assignment page on Canvas by **11:59pm on Sunday, May 2**.
6. Late submissions will be accepted up to 24 hours after the deadline, however they will be penalized: 4 points off for every six hours the submission is late.

Portion	Question	Points	Score
Theory	1	20	
	2	20	
Computing	1	20	
	2	20	
	3	20	
Total		100	

Exam 2

Jeff Gould

4/30/2021

Theoretical

1

a

$$P(s_i) = \left(\frac{\lambda}{2\pi s_i^3}\right)^{1/2} \exp\left[-\frac{\lambda(s_i-\mu)^2}{2\mu^2 s_i}\right]$$

$$\begin{aligned}
\mathcal{L}(s_i|\mu,\lambda) &= \prod \left(\frac{\lambda}{2\pi s_i^3}\right)^{1/2} \exp\left[-\frac{\lambda(s_i-\mu)^2}{2\mu^2 s_i}\right] \\
&\propto \lambda^{n/2} \exp\left[-\sum \frac{\lambda(s_i-\mu)^2}{2\mu^2 s_i}\right] \\
&= \lambda^{n/2} \exp\left[-\lambda \sum \frac{(s_i-\mu)^2}{2\mu^2 s_i}\right]
\end{aligned}$$

We recognize this as the kernel for a random variable with a gamma distribution. So a conjugate prior for lambda would be

$$\pi(\lambda) \propto \lambda^{\alpha-1} e^{-\beta\lambda}$$

Then the conditional posterior for λ becomes:

$$\begin{aligned}
P(\lambda|\mu,s_i) &\propto \mathcal{L}(s_i|\lambda,\mu)\pi(\lambda) \\
&\propto \lambda^{n/2} \exp\left[-\lambda \sum \frac{(s_i-\mu)^2}{2\mu^2 s_i}\right] \lambda^{\alpha-1} e^{-\beta\lambda} \\
&= \lambda^{\alpha+n/2-1} \exp\left[-\lambda\left(\beta + \sum \frac{(s_i-\mu)^2}{2\mu^2 s_i}\right)\right]
\end{aligned}$$

This is the kernel for $\lambda \sim IG\left(\alpha + n/2, \beta + \sum \frac{(s_i-\mu)^2}{2\mu^2 s_i}\right)$

b

Define $\phi = \mu^{-1}$

$$\begin{aligned}
P(s_i|\lambda, \phi = 1/\mu) &= \left(\frac{\lambda}{2\pi s_i^3}\right)^{1/2} \exp\left[-\frac{\lambda(s_i - 1/\phi)^2}{2(1/\phi)^2 s_i}\right] \\
&= \left(\frac{\lambda}{2\pi s_i^3}\right)^{1/2} \exp\left[-\frac{\lambda}{2s_i}\phi^2(s_i^2 - 2s_i(1/\phi) + (1/\phi)^2)\right] \\
&= \left(\frac{\lambda}{2\pi s_i^3}\right)^{1/2} \exp\left[-\frac{\lambda}{2s_i}\left(\phi^2 s_i^2 - 2s_i\phi + 1\right)\right] \\
&= \left(\frac{\lambda}{2\pi s_i^3}\right)^{1/2} \exp\left[-\frac{\lambda s_i^2}{2s_i}(\phi^2 - 2\phi/s_i + 1/s_i^2)\right] \\
&= \left(\frac{\lambda}{2\pi s_i^3}\right)^{1/2} \exp\left[-\frac{\lambda s_i}{2}\left(\phi - \frac{1}{s_i}\right)^2\right]
\end{aligned}$$

Now find the posterior:

$$\begin{aligned}
P(\phi|\lambda, S) &= \prod \left(\frac{\lambda}{2\pi s_i^3}\right)^{1/2} \exp\left[-\frac{\lambda s_i}{2}\left(\phi - \frac{1}{s_i}\right)^2\right] \\
&\propto \exp\left[-\sum \frac{\lambda s_i}{2}\left(\phi - \frac{1}{s_i}\right)^2\right] \\
&= \exp\left[-\frac{\lambda}{2}\sum s_i\left(\phi - \frac{1}{s_i}^2\right)\right] \\
&= \exp\left[-\frac{\lambda}{2}\sum s_i\left(\phi - \frac{1}{\bar{s}} + \frac{1}{\bar{s}} - \frac{1}{s_i}\right)^2\right] \\
&= \exp\left[-\frac{\lambda}{2}\sum s_i\left((\phi - \bar{s}^{-1})^2 + 2(\phi - \bar{s}^{-1})(\bar{s}^{-1} - s_i^{-1}) + (\bar{s}^{-1} - s_i^{-1})^2\right)\right] \\
&= \exp\left[-\frac{\lambda}{2}\left(\sum s_i(\phi - \bar{s}^{-1})^2 + \sum 2(\phi - \bar{s}^{-1})(\bar{s}^{-1} - s_i^{-1}) + \sum(\bar{s}^{-1} - s_i^{-1})^2\right)\right] \\
&= \exp\left[-\frac{\lambda}{2}\left(\sum s_i(\phi - \bar{s}^{-1})^2 + \sum(\bar{s}^{-1} - s_i^{-1})^2\right)\right] \\
&\propto \exp\left[-\frac{\lambda}{2}\left(\sum s_i(\phi - \bar{s}^{-1})^2\right)\right] \\
&= \exp\left[-\frac{n\lambda\bar{s}}{2}\left(\phi - \bar{s}^{-1}\right)^2\right]
\end{aligned}$$

Which is the kernel for a normally distributed random variable centered at $\bar{s}^{-1}$ with variance $\sigma^2 = (n\lambda\bar{s})^{-1}$

2

a

Rejection sampling is useful with lower dimensional data and when it is relatively easy to find a proposal distribution that encompasses the underlying density. Sometimes finding a good proposal density can be

difficult, or to find one that fits will require a lot of rejections to properly cover the distribution. Another drawback is that it does not scale to multiple parameters very well, and the computation time will be extensive. Benefits are that a proper rejection sampler will converge to the underlying distribution, and all observations are *iid*

The Metropolis Algorithm requires a symmetric proposal density (but the MH-Algorithm easily fixes this). Benefits are that it scales better to higher dimensions, less information about the underlying density needs to be known (ie, up to a constant of proportionality). One of the drawbacks of the Metropolis Algorithm is that the draws are correlated, ie each $\theta^{(b)}$ is correlated with $\theta^{(b-1)}$, and we may need to increase our sample time in order to thin out the correlation between samples, increasing our run time.

b

We have starting value $\theta^{(0)} = 2.5$. We then draw $\theta^* = -10$. We then calculate $r = P(\theta^*)/P(\theta^{(0)}) \approx 0$, as the probabilty that $\theta = -10$ looks to be equal to 0. We sample $U \sim Unif(0,1)$, and if $U > \min(r,1)$, then define $\theta^{(1)} = \theta^{(0)}$, else $\theta^{(1)} = \theta^*$

Now we draw $\theta^* = 5$. We get $r = P(\theta^*)/P(\theta^{(1)}) > 1$, since the density is greater at 5 than 2.5 (and -10).

We sample $U \sim Unif(0,1)$, and since $U < \min(r,1) = 1$, $\theta^{(2)} = \theta^* = 5$

Now we draw $\theta^* = 0$. $r = P(\theta^* = 0)/P(\theta^{(2)} = 5) \approx 0.275/0.2 > 1$

Draw $U \sim Unif(0,1)$. We will get $U < \min(r,1) = 1$, so define $\theta^{(3)} = \theta^* = 0$

Now we draw $\theta^* = 10$

Define $r = P(\theta^* = 10)/P(\theta^{(3)} = 0) \approx 0.01/0.275$

Draw $U \sim Unif(0,1)$. If $U < \min(r,1) = r$, then define $\theta^{(4)} = 10$. Else, $\theta^{(4)} = \theta^{(3)} = 0$

$b_1 \in (2.5, -10)$

$b_2 \in (5)$

$b_3 \in (0)$

$b_4 \in (0, 10)$

with our most likely combination $(b_1, b_2, b_3, b_4) = (2.5, 5, 0, 0)$

Computation

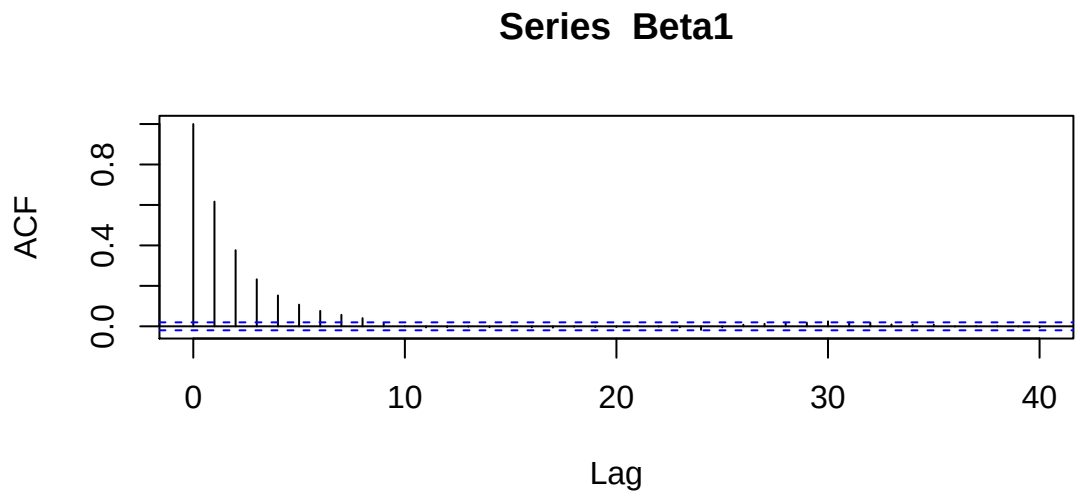
1

$$\begin{aligned}
\mathcal{L} &= \prod \frac{\beta}{2\Gamma(1/\beta)} \exp(-|z_i|^\beta) \\
P(\beta|Z) &= \prod \frac{\beta}{2\Gamma(1/\beta)} \exp(-|z_i|^\beta) \\
&= \left(\frac{\beta}{2\Gamma(1/\beta)}\right)^n \exp\left(-\sum |z_i|^\beta\right) \\
&\propto \frac{\beta^n}{\Gamma(1/\beta)^n} \exp\left(-\sum |z_i|^\beta\right)
\end{aligned}$$

```r
coup <- read.table("coup1280.txt", sep = " ", header = T)
Z <- (coup$logCoup - mean(coup$logCoup)) / sd(coup$logCoup)

P_beta <- function(beta, z = Z){
  n <- length(z)

  P_b <- beta^n / gamma(1/beta)^n * exp(- sum( abs(z)^beta))
  return(P_b)
}
B <- 16000


MH_algo <- function(seed, beta_1, B, thin = 1, data = Z){
  set.seed(seed)

  Beta <- vector("numeric", B)
  accept <- 0
  Beta[1] = beta_1

  for (b in 2:B) {

    beta_star <- statmod::rinvgauss(1, mean = 1.5, 25)


    r <- (P_beta(beta_star, data) / statmod::dinvgauss(beta_star, 1.5, 25) ) /
      (P_beta(Beta[b-1], data) / statmod::dinvgauss(Beta[b-1], 1.5, 25))

    U <- runif(1)

    if(U <= min(r, 1)){
      accept <- accept + 1
      beta_1 <- beta_star
    }

    Beta[b] <- beta_1

  }
  print(glue::glue("Acceptance Rate: {accept / B}"))
  Beta <- Beta[-(1:(B/2))]
  print(glue::glue("Geweke Diagnostic: {coda::geweke.diag(Beta)}"))
  Beta <- Beta[seq(1, B/2, by = thin)]
  return(Beta)

}

Beta1 <- MH_algo(37, 1.25, 20000, thin = 1)
```

```
## Acceptance Rate: 0.3031
## Geweke Diagnostic: c(var1 = -1.80573393709298)
## Geweke Diagnostic: c(0.1, 0.5)
```

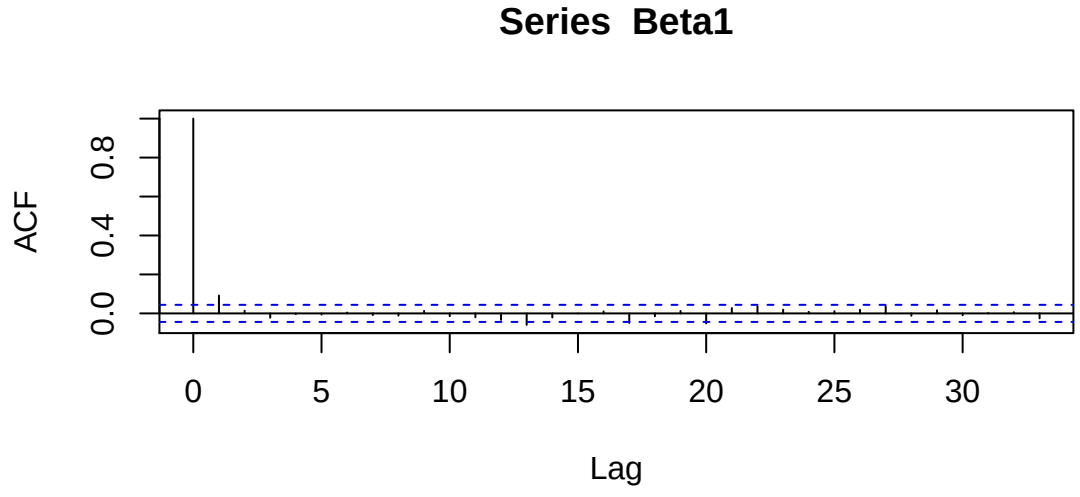
```r
acf(Beta1)
```




```r
Beta1 <- MH_algo(37, 1.25, 20000, thin = 5)
```

```
## Acceptance Rate: 0.3031
## Geweke Diagnostic: c(var1 = -1.80573393709298)
## Geweke Diagnostic: c(0.1, 0.5)
```

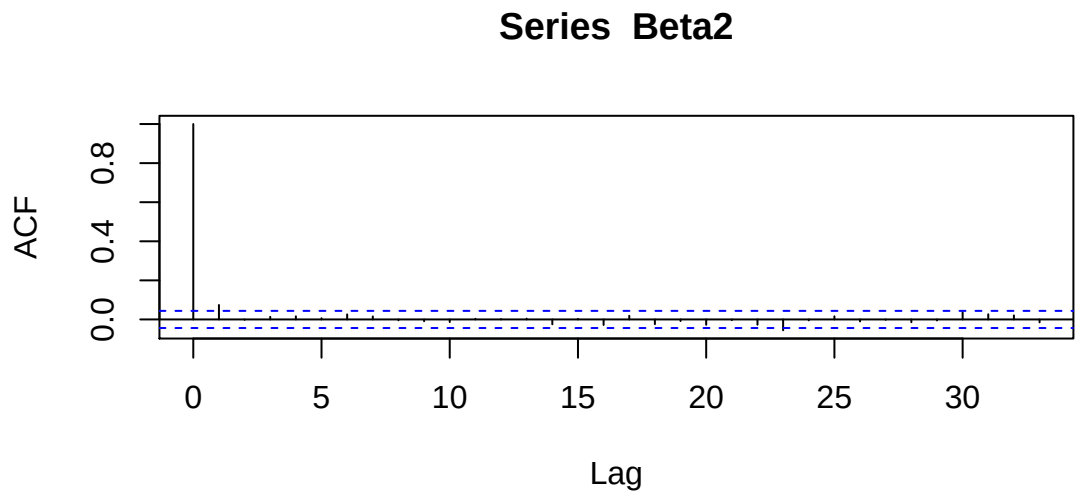
```r
acf(Beta1)
```




```r
Beta2 <- MH_algo(181, 1, 20000, thin = 5)
```

```
## Acceptance Rate: 0.2965
## Geweke Diagnostic: c(var1 = 1.27638054165452)
## Geweke Diagnostic: c(0.1, 0.5)
```

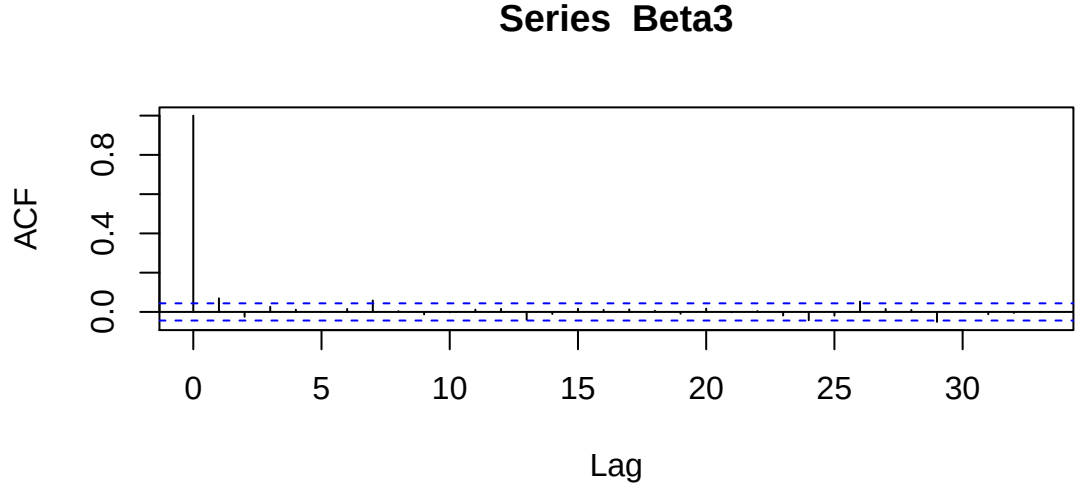
```r
acf(Beta2)
```




```r
Beta3 <- MH_algo(142, 1.5, 20000, thin = 5)
```

```
## Acceptance Rate: 0.29935
## Geweke Diagnostic: c(var1 = 1.7907160612524)
## Geweke Diagnostic: c(0.1, 0.5)
```

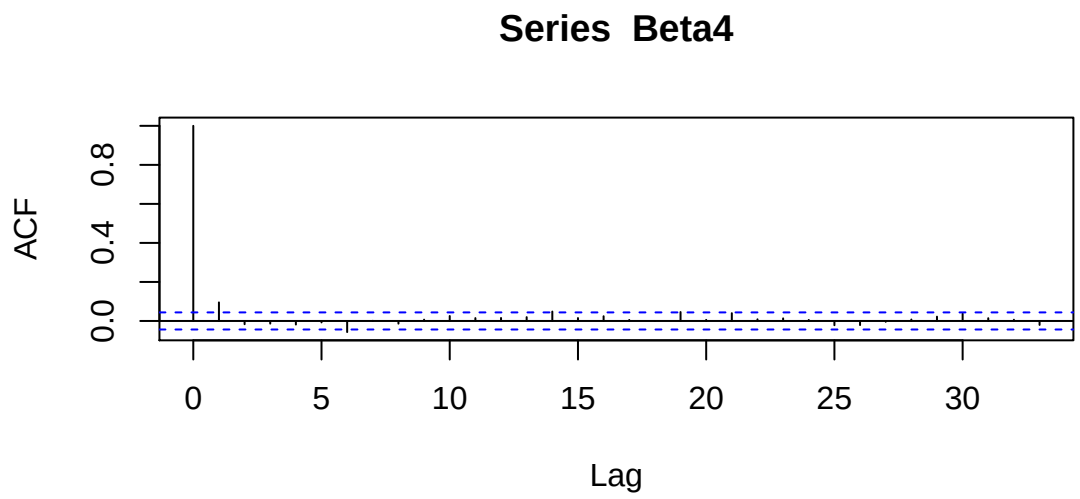
```r
acf(Beta3)
```




```r
Beta4 <- MH_algo(1004, 2, 20000, thin = 5)
```

```
## Acceptance Rate: 0.29695
## Geweke Diagnostic: c(var1 = 1.58216766250913)
## Geweke Diagnostic: c(0.1, 0.5)
```

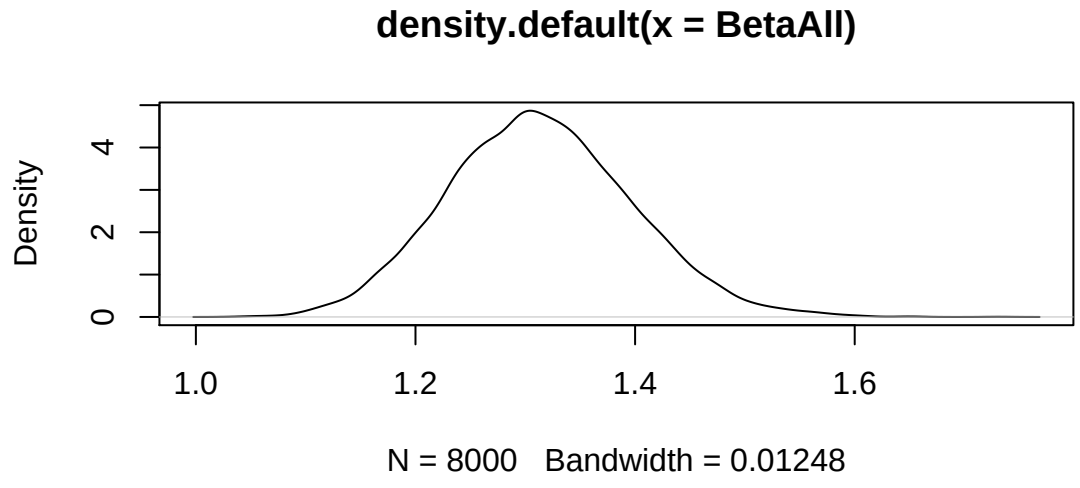
```r
acf(Beta4)
```




```r
BetaAll <- c(Beta1, Beta2, Beta3, Beta4)
mean(BetaAll)
```

```
## [1] 1.313582
```

```r
plot(density(BetaAll))
```



Since $\beta = 1.3135816$ is closer to 1 than 2, we conclude that a Laplacian distribution is better suited for our dataset, giving $P(x|\mu, b) = \frac{1}{2b} \exp\left[-\frac{|x-\mu|}{b}\right]$

Set $\pi(\mu) \propto 1$ and $\pi(b) \propto b^{-1}$

Define $\bar{x}$ to be the *median* for our data, as the median is the MLE for μ of a Laplace distribution

$$
\begin{aligned}
P(\mu|X,b) &\propto \prod \exp\left[-\frac{|x_i-\mu|}{b}\right] \\
&= \exp\left[-\frac{1}{b}\sum|x_i-\mu|\right] \\
&= \exp\left[-\frac{1}{b}\sum|\mu-x_i|\right] \\
&= \exp\left[-\frac{1}{b}\sum|(\mu-\bar{x})-(x_i-\bar{x})|\right] \\
&= \exp\left[-\frac{1}{b}\sum\{1_{x_i\leq\mu}((\mu-\bar{x})-(x_i-\bar{x}))+1_{x_i>\mu}((x_i-\bar{x})-(\mu-\bar{x}))\}\right] \\
&= \exp\left[-\frac{1}{b}\left(\sum\{1_{x_i\leq\mu}((\mu-\bar{x})-(x_i-\bar{x}))\}+\sum\{1_{x_i>\mu}((x_i-\bar{x})-(\mu-\bar{x}))\}\right)\right] \\
&= \exp\left[-\frac{1}{b}\left(\sum\{1_{x_i\leq\mu}(\mu-\bar{x})-1_{x_i>\mu}(\mu-\bar{x})\}+\sum\{1_{x_i>\mu}(x_i-\bar{x})-1_{x_i\leq\mu}(x_i-\bar{x})\}\right)\right] \\
&\propto \exp\left[-\frac{1}{b}\left(\sum\{1_{x_i\leq\mu}(\mu-\bar{x})-1_{x_i>\mu}(\mu-\bar{x})\}\right)\right] \\
&= \exp\left[-\frac{1}{b}\sum|\mu-\bar{x}|\right] \\
&= \exp\left[-\frac{1}{b/n}|\mu-\bar{x}|\right]
\end{aligned}
$$

So the posterior conditional distribution for μ is a Laplace Distribution with location $= \bar{x}$ and scale $= \frac{b}{n}$

$\pi(b) \propto (b)^{-1}$

$$
\begin{aligned}
P(b|\mu,X) &\propto b^{-1}\prod\frac{1}{2b}\exp\left[-\frac{|x_i-\mu|}{b}\right] \\
&\propto b^{-(n+1)}\exp\left(-\frac{\sum|x_i-\mu|}{b}\right)
\end{aligned}
$$

The kernel for an inverse-gamma random variable with $\alpha = n$ and $\beta = \sum|x_i-\mu|$

Set up Gibbs Sampler:

```
B <- 2*10000
X <- coup$logCoup
n <- length(X)
b <- vector("numeric", B)
mu <- vector("numeric", B)

b[1] = sd(X)^2
mu[1] = median(X)

x_bar <- median(X)
set.seed(123)
for (i in 2:B) {

  scale <- b[i-1] / n
```

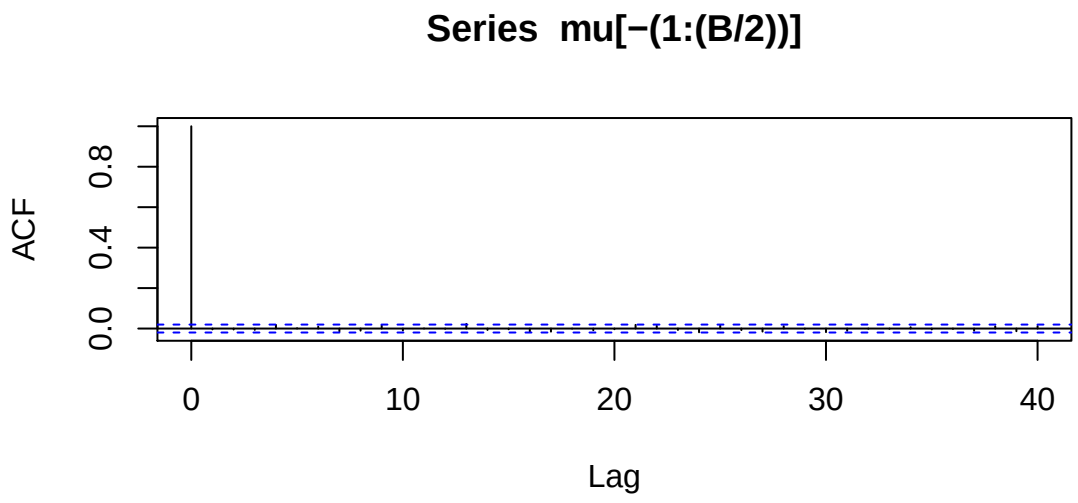
```
  alpha <- n
  beta <- sum(abs(X - mu[i-1]))

  mu[i] <- extraDistr::rlaplace(1, x_bar, scale)
  b[i] <- MCMCpack::rinvgamma(1, alpha, beta)

}
coda::geweke.diag(mu)
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##    var1
## -0.6474
```

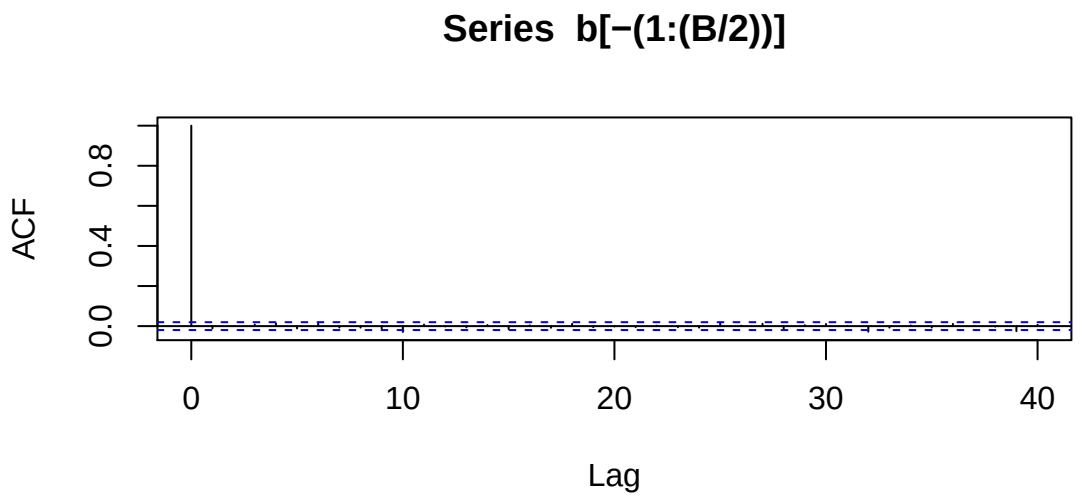
```
coda::geweke.diag(b)
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##  var1
## 1.326
```

```
acf(mu[-(1:(B/2))])
```



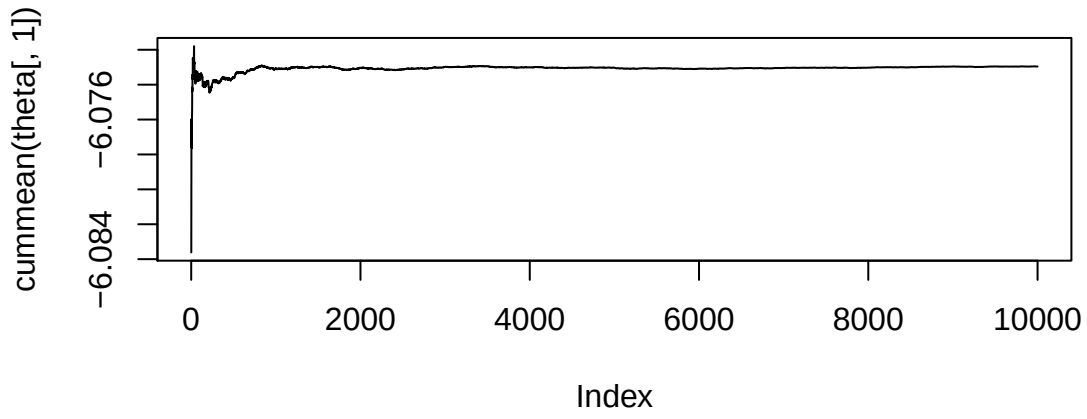

```
acf(b[-(1:(B/2))])
```



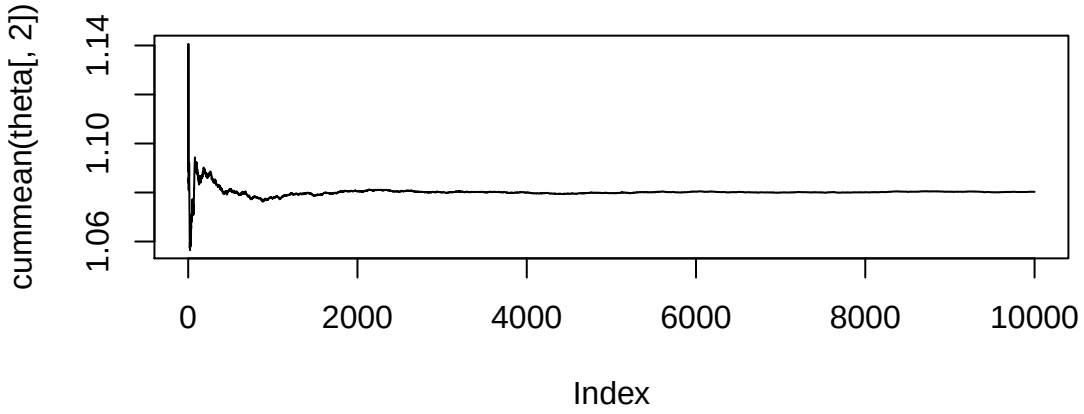

```
theta <- matrix(c(mu[-(1:(B/2))],
                  b[-(1:(B/2))]), ncol = 2)

coefs <- apply(theta, 2, quantile, probs = c(0.5, 0.025, 0.975))
colnames(coefs) <- c("mu", "b")
round(t(coefs), 4)
```

```
##        50%    2.5%   97.5%
## mu -6.0730 -6.0924 -6.0541
## b   1.0766  0.9274  1.2578
```

```
plot(cummean(theta[,1]), type = "l")
```



```r
plot(cummean(theta[,2]), type = "l")
```



2 Veteran Administration

$$P(t_i|\theta,\lambda) = \frac{\theta}{\lambda^\theta} t_i^{\theta-1} \exp\left[-\left(\frac{t_i}{\lambda}\right)^\theta\right]$$

$$\pi(\lambda,\theta) \propto (\lambda^\theta)^{-1}$$

$$\begin{aligned} P(\lambda,\theta|t) &\propto (\lambda^\theta)^{-1} \prod \frac{\theta}{\lambda^\theta} t_i^{\theta-1} \exp\left[-\left(\frac{t_i}{\lambda}\right)^\theta\right] \\ &\propto (\lambda^\theta)^{-1} \left(\frac{\theta}{\lambda^\theta}\right)^n \exp\left[-\sum\left(\frac{t_i}{\lambda}\right)^\theta\right] \prod t_i^\theta \\ &= \frac{\theta^n}{(\lambda^\theta)^{n+1}} \exp\left[-\frac{1}{\lambda^\theta}\sum t_i^\theta\right] \prod t_i^\theta \end{aligned}$$

$$P(\lambda|\theta,t) \propto (\lambda^\theta)^{-(n+1)} \exp\left[-\frac{1}{\lambda^\theta}\sum t_i^\theta\right]$$

Substitute a for λ^θ, and we see this is the kernel for an inverse-gamma distributed r.v., with parameters $\alpha = n$ and $\beta = \sum t_i^\theta$

$$a \sim IG(n, \sum t_i^\theta)$$

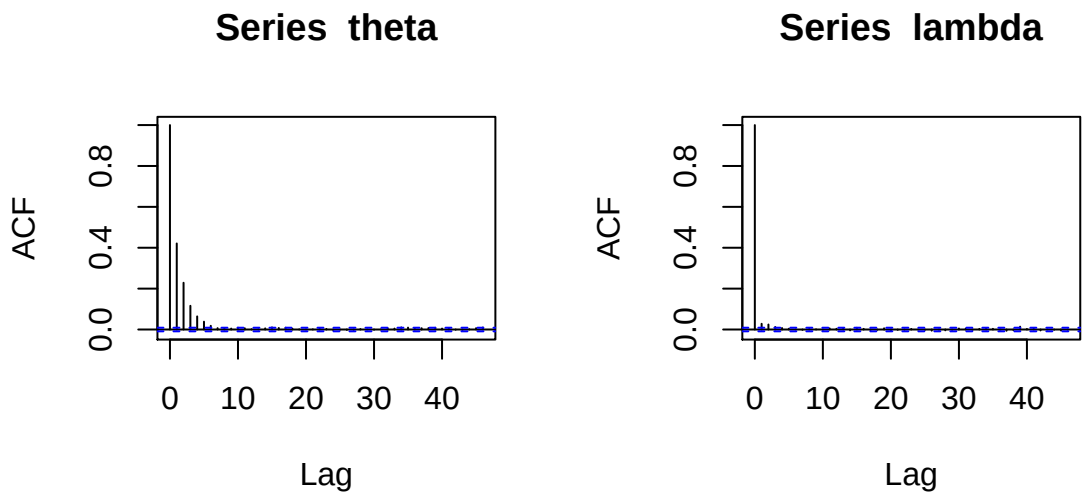
$$P(\theta|\lambda,t_i) \propto \frac{\theta^n}{(\lambda^\theta)^{n+1}} \exp\left[-\frac{1}{\lambda^\theta}\sum t_i^\theta\right] \prod t_i^\theta$$

Which has no reduceable form

```r
valc <- read.table("valc.txt", header = T)

t <- valc$t

p_theta <- function(theta, lambda, tt){

  n <- length(tt)
  return(
    theta^n / (lambda^theta)^(n+1) * exp(-1/(lambda^theta) * sum(tt^theta)) * prod(tt^theta)
  )
}

GibbsSample <- function(theta_init, lambda_init, alpha, beta, seed, B = 50000){

  theta = vector("numeric", B)
  lambda = vector("numeric", B)

  lambda[1]  <- lambda_init
  theta[1] <- theta_init
  set.seed(seed)
  n <- length(t)

  alpha = 4
  beta = 3
  accept <- 0
  for(b in 2:B){

    a <- lambda[b-1]^theta[b-1]
    a <- MCMCpack::rinvgamma(1, n, sum(t^theta[b-1]))
    lambda[b] <- a^(1 / theta[b-1])

    theta_star <- rgamma(1, alpha, beta)

    r <- (p_theta(theta_star, lambda[b-1], t) / dgamma(theta_star, alpha, beta) ) /
      p_theta(theta[b-1], lambda[b-1], t) / dgamma(theta[b-1], alpha, beta)

    while(is.nan(r)){
      theta_star <- rgamma(1, alpha, beta)

      r <- (p_theta(theta_star, lambda[b-1], t) / dgamma(theta_star, alpha, beta) ) /
        p_theta(theta[b-1], lambda[b-1], t) / dgamma(theta[b-1], alpha, beta)

    }
    U <- runif(1)

    if(U < r){
      accept <- accept + 1
      theta_init <- theta_star
    }
    theta[b] <- theta_init
  }
  print(glue::glue("Our MH acceptance rate is: {accept / B}"))
```

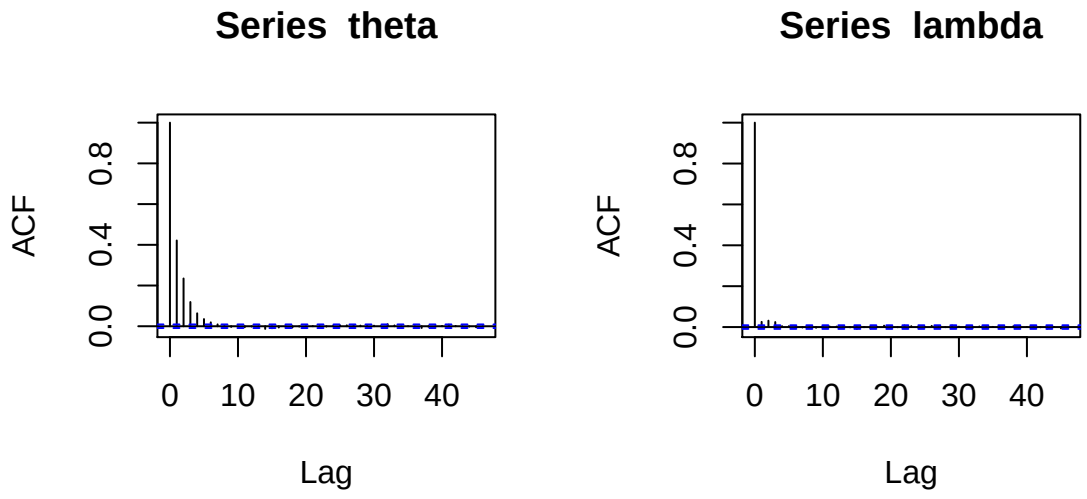
```
  acf(theta)
  acf(lambda)

  return(list(lambda = lambda,
              theta = theta))
}


GS1 <- GibbsSample(theta_init = 0.1, lambda_init = 1, seed = 121, alpha = 4.5, beta = 3.5)
```

```
## Our MH acceptance rate is: 0.4936
```



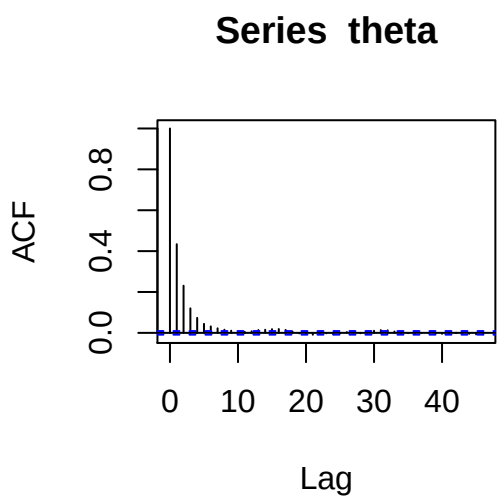

```
GS2 <- GibbsSample(theta_init = 0.2, lambda_init = 2, seed = 75, alpha = 4.5, beta = 3.5)
```

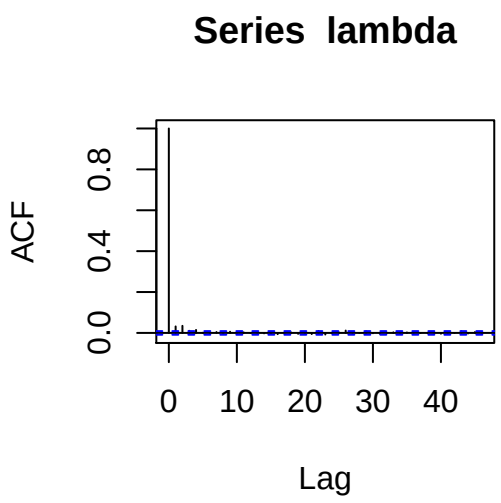
```
## Our MH acceptance rate is: 0.49124
```



```
GS3 <- GibbsSample(theta_init = 0.15, lambda_init = 0.5, seed = 340, alpha = 4.5, beta = 3.5)
```

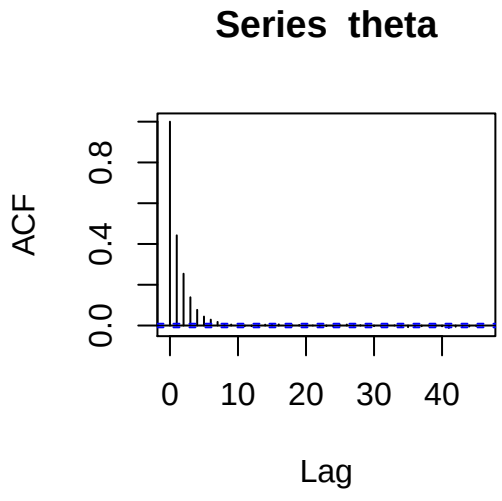
```
## Our MH acceptance rate is: 0.48838
```

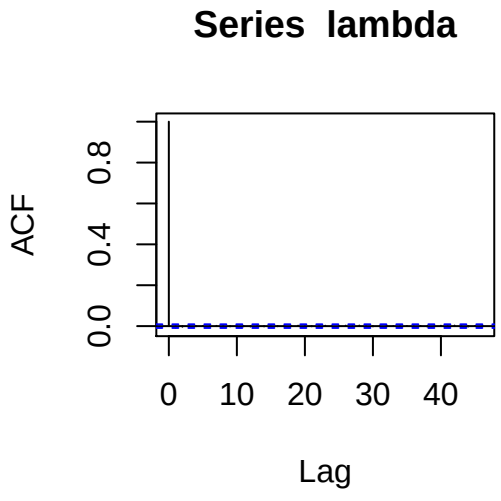





```
GS4 <- GibbsSample(theta_init = 0.05, lambda_init = 1.5, seed = 19, alpha = 4.5, beta = 3.5)
```

```
## Our MH acceptance rate is: 0.4872
```

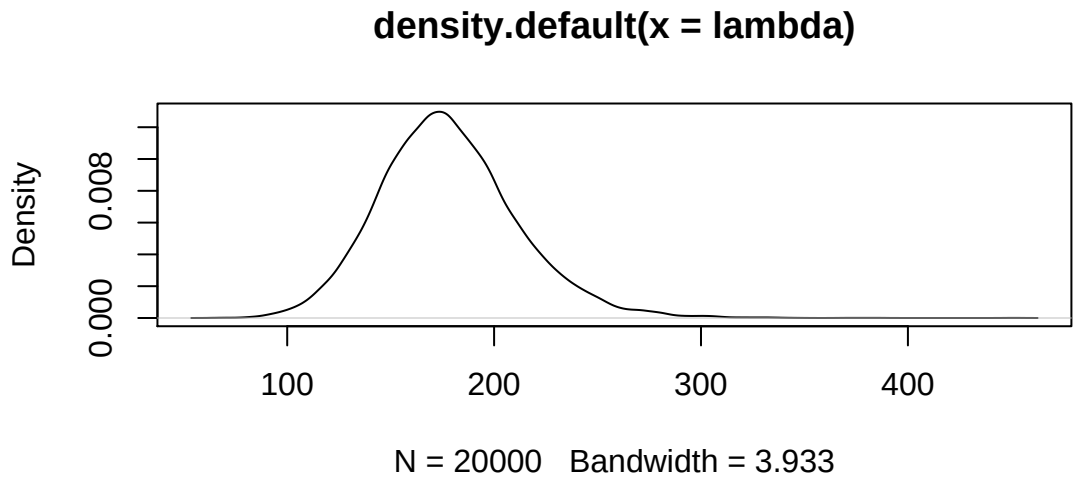




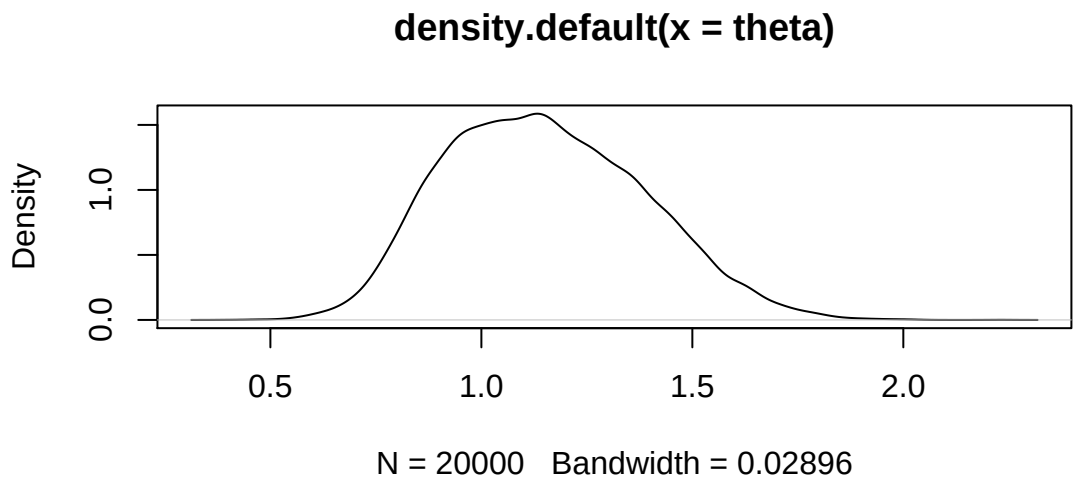
Thin to one out of every 5 observations

```
B <- 50000
thin <- 5

theta <- c(
```

```
  GS1$theta[seq(B/2+1, B, thin)],
  GS2$theta[seq(B/2+1, B, thin)],
  GS3$theta[seq(B/2+1, B, thin)],
  GS4$theta[seq(B/2+1, B, thin)]
)

lambda <- c(
  GS1$lambda[seq(B/2+1, B, thin)],
  GS2$lambda[seq(B/2+1, B, thin)],
  GS3$lambda[seq(B/2+1, B, thin)],
  GS4$lambda[seq(B/2+1, B, thin)]
)

plot(density(lambda))
```




```
plot(density(theta))
```



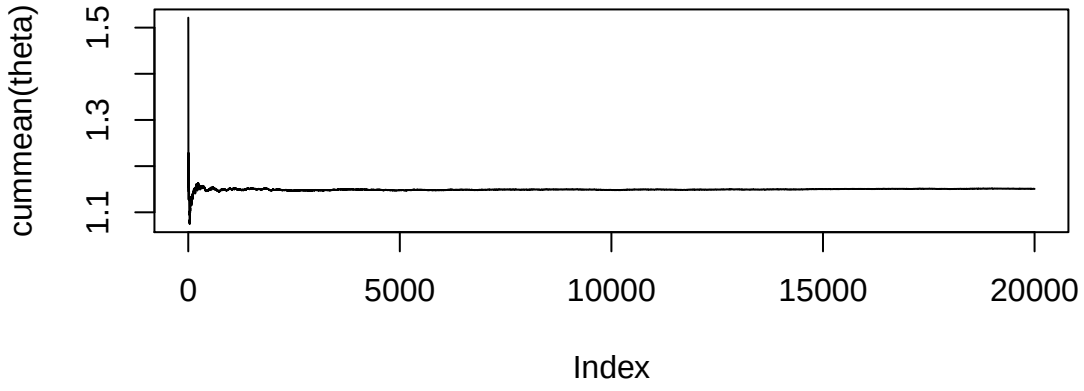

```
Lambda <- mean(lambda)
Theta <- mean(theta)

coda::geweke.diag(theta)
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##    var1
## -0.6759
```

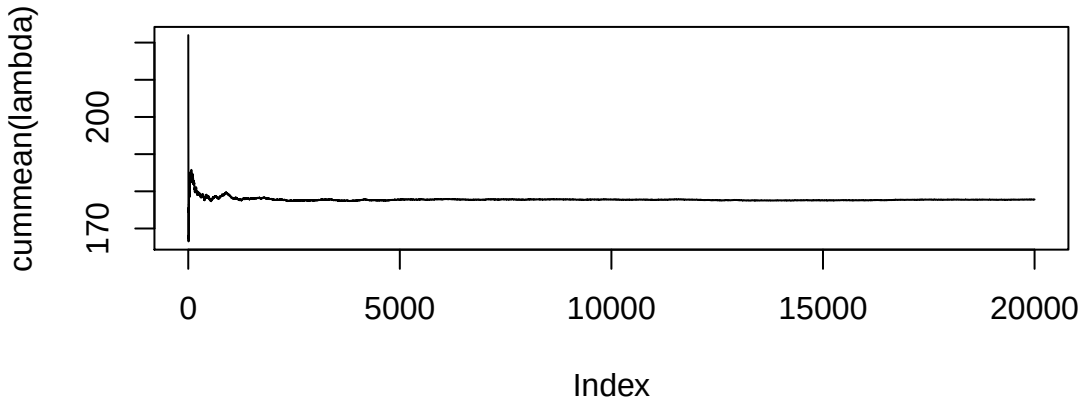
```
coda::geweke.diag(lambda)
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##     var1
## -0.08202
```

```
plot(cummean(theta), type = "l")
```

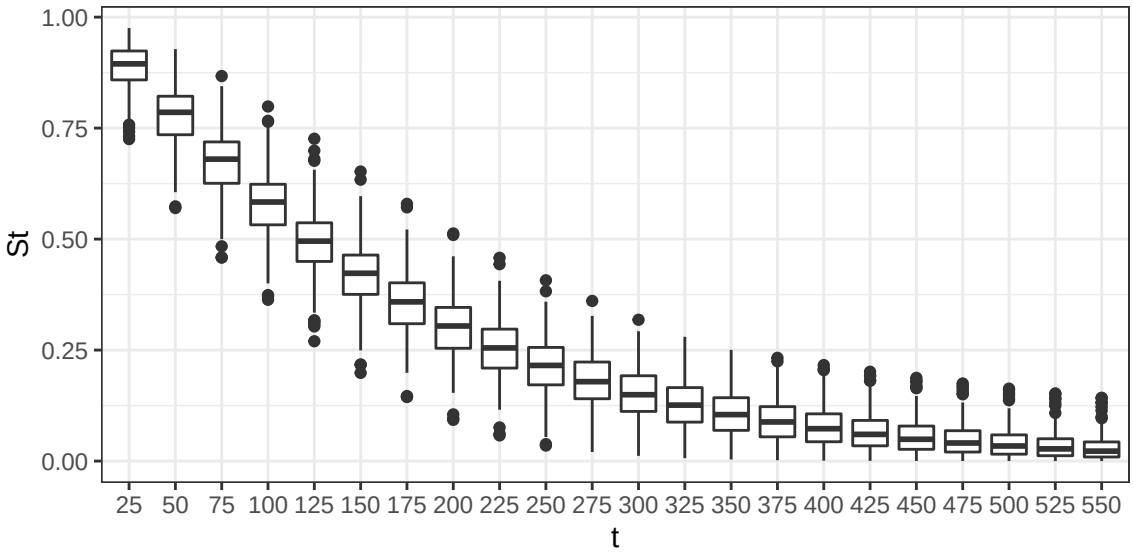



```
plot(cummean(lambda), type = "l")
```



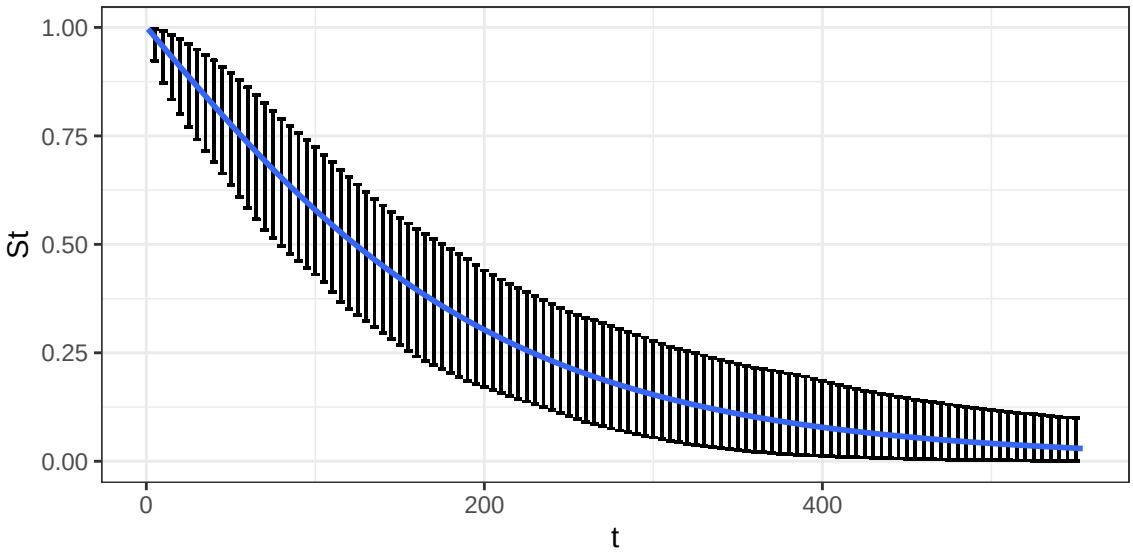

```
S_t <- function(t, theta, lambda){
  St <- 1 - pweibull(t, shape = theta, scale = lambda)
  return(data.frame(t = t, St = St))
}

t_grid <- seq(1, max(valc$t)+1, 1)

i <- sample(x = 1:length(lambda), size = 250, replace = F)


SurvivalDist <- data.frame(t = rep(t_grid, 251),
           lambda = c(Lambda, lambda[i]),
           theta = c(Theta, theta[i])) %>% pmap_dfr(.f = S_t)
```

```
SurvivalDist %>%
  filter(t %in% seq(0, 600, 25)) %>%
  mutate(t = as.factor(t)) %>%
  #mutate(avgParams = case_when(lambda == Lambda ~ 1, TRUE ~0)) %>%
  ggplot(aes(x = t, y = St)) +
  geom_boxplot() +
  theme_bw()
```




```
SurvivalDist %>%
  filter(t %in% seq(0, 600, 5)) %>%
  ungroup() %>%
  group_by(t) %>%
  mutate(upper = quantile(St, 0.975),
         lower = quantile(St, 0.025)) %>%
  ggplot(aes(x = t, y = St, ymin = lower, ymax = upper)) +
  geom_errorbar() +
  geom_smooth(data = SurvivalDist, aes(x = t, y = St), inherit.aes = F) +
  theme_bw()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



3

The distribution for a Bernoulli r.v. is $P(x|p) = p^x(1-p)^{1-x}$

Given coefficients β, then for each wine we get $\hat{p}_i = \frac{exp(\beta \cdot x_i)}{1+\exp(\beta \cdot x_i)}$

Our log-likelihood becomes:

$$\begin{aligned} \log \mathcal{L} &= \log \prod \hat{p}_i^{y_i}(1-\hat{p}_i)^{1-y_i} \\ &= \sum \log \left(\hat{p}_i^{y_i}(1-\hat{p}_i)^{1-y_i}\right) \end{aligned}$$

For the flat prior, $\pi(\beta) \propto 1$, this is our posterior

```
vino <- read.table("vinhoverde.txt", header = T) %>% select(quality, pH, sulphates, alcohol)

Y <- vino$quality
n <- length(Y)

X <- model.matrix(quality ~ pH + sulphates + alcohol, data = vino)
fit     <- glm(quality ~ pH + sulphates + alcohol,
               data = vino,
               family = binomial(link = 'logit'))

bhat    <- coef(fit)
vbeta   <- vcov(fit)
B         <- 2*4000

beta        <- matrix(0, nrow = B, ncol = ncol(X))
ar          <- vector('numeric', length = B)

beta[1,]    <- bhat # posterior mode
```

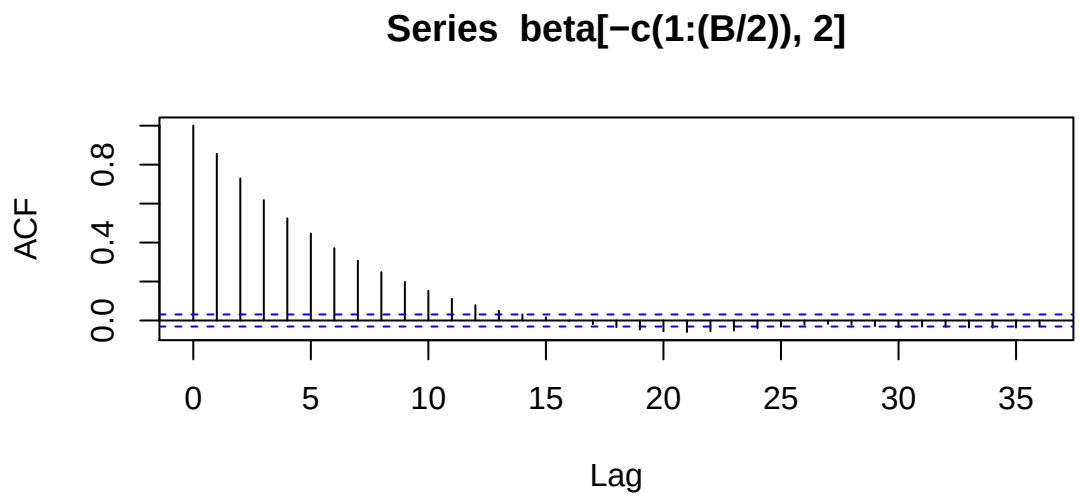
```r
logL <- function(b,  x, y){
  p_hat <- exp(x %*% b) / (1 + exp(x %*% b))
  lL  <- sum(
    log(
      p_hat^y * (1 - p_hat)^(1 - y)
    )
  )
  return(lL)
}

tau <- 1.5 # need to tune this

set.seed(870)
for(t in 2:B){

  bstar <- mvtnorm::rmvnorm(1, beta[t-1,], tau*vbeta)
  r     <- exp(logL(t(bstar), X, Y)  -  logL(beta[t-1,], X, Y) )
  U     <- runif(1)

  if(U < min(1,r)){
    beta[t,]    <- bstar
    ar[t]         <- 1
  } else{
    beta[t,]    <- beta[t-1,]
    ar[t]         <- 0
  }

}

mean(ar[-c(1:(B/2))])
```

```
## [1] 0.29775
```

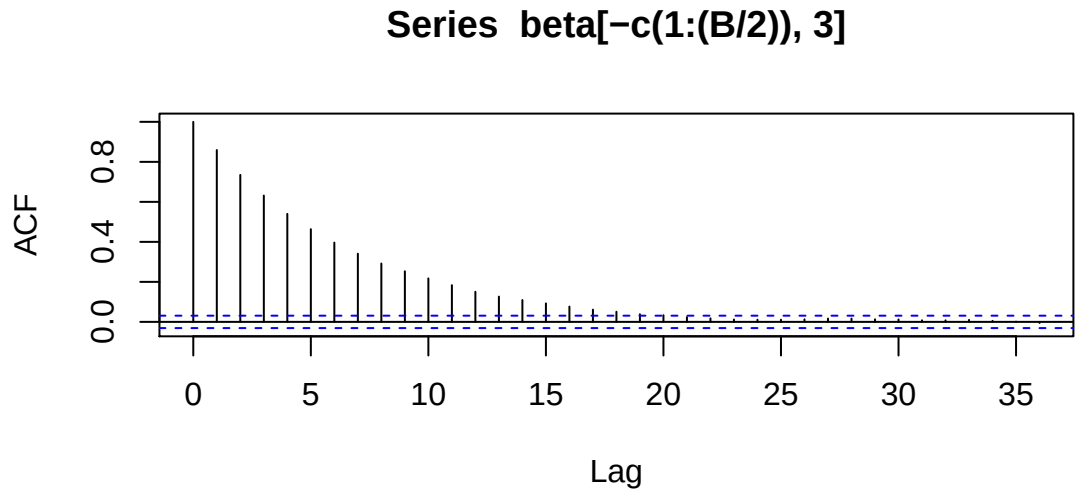
```r
# coefs <- apply(beta[-c(1:B/2),], 2, quantile, probs = c(0.5, 0.025, 0.975))
# colnames(coefs) <- colnames(X)
# round(t(coefs), 4)
# coef(fit)
coda::geweke.diag(beta[-c(1:(B/2)),])
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##    var1    var2    var3    var4
## -0.8286 -0.9156 -0.1616  3.0640
```

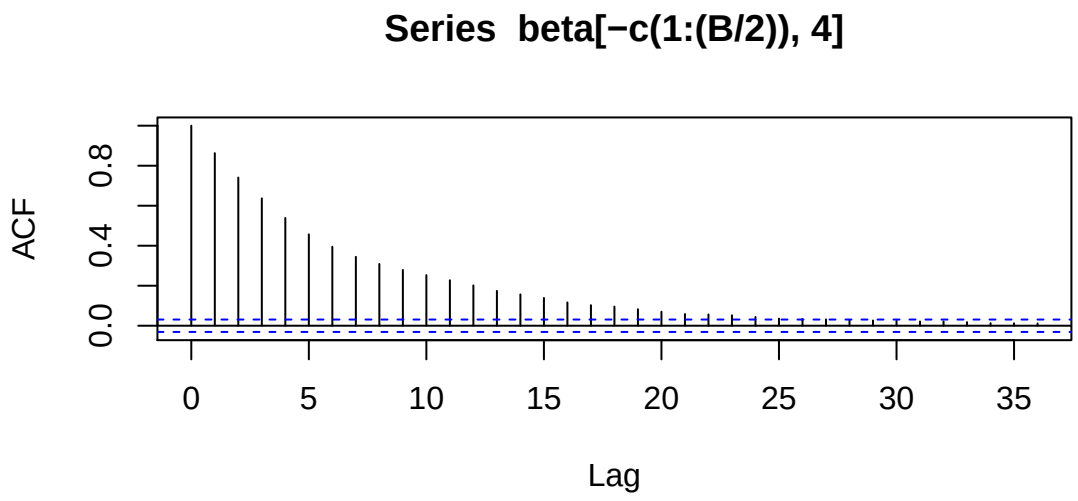
```r
acf(beta[-c(1:(B/2)), 2])
```




```
acf(beta[-c(1:(B/2)), 3])
```




```
acf(beta[-c(1:(B/2)), 4])
```




```
#mcmcplots::mcmcplot1(beta)
```

Let's run it again, this time with thinning every 25 samples

```
burnIn <- 10000
thin <- 25
B         <- burnIn + 4000*thin # burn-in + keep * thin

beta        <- matrix(0, nrow = B, ncol = ncol(X))
ar          <- vector('numeric', length = B)

beta[1,]    <- bhat # posterior mode

tau <- 1.5 # need to tune this

set.seed(870)
for(t in 2:B){

  bstar <- mvtnorm::rmvnorm(1, beta[t-1,], tau*vbeta)
  r     <- exp(logL(t(bstar), X, Y)  -  logL(beta[t-1,], X, Y) )
  U     <- runif(1)

  if(U < min(1,r)){
    beta[t,]    <- bstar
    ar[t]         <- 1
  } else{
    beta[t,]    <- beta[t-1,]
    ar[t]         <- 0
  }

}

mean(ar[-c(1:burnIn)])
```

```
## [1] 0.29215
```

```r
p_not0 <- function(bb){
  pointEst <- mean(bb)
  if(pointEst > 0){
    pp <- mean(bb >0)
  }else{
    pp <- mean(bb < 0)
  }
  return(pp)
}

BETA <- beta[seq(burnIn + 1, B, thin), ]
coefs <- apply(BETA, 2, quantile, probs = c(0.5, 0.05, 0.95))
colnames(coefs) <- colnames(X)
round(t(coefs), 4)
```

```
##                50%       5%     95%
## (Intercept) -9.4533 -10.7658 -8.2145
## pH           0.3229  -0.0506  0.7000
## sulphates    1.3116   0.7962  1.8277
## alcohol      0.8240   0.7676  0.8791
```

```r
coef(fit)
```

```
## (Intercept)          pH   sulphates     alcohol
##  -9.4447477   0.3175969   1.3055717   0.8232929
```

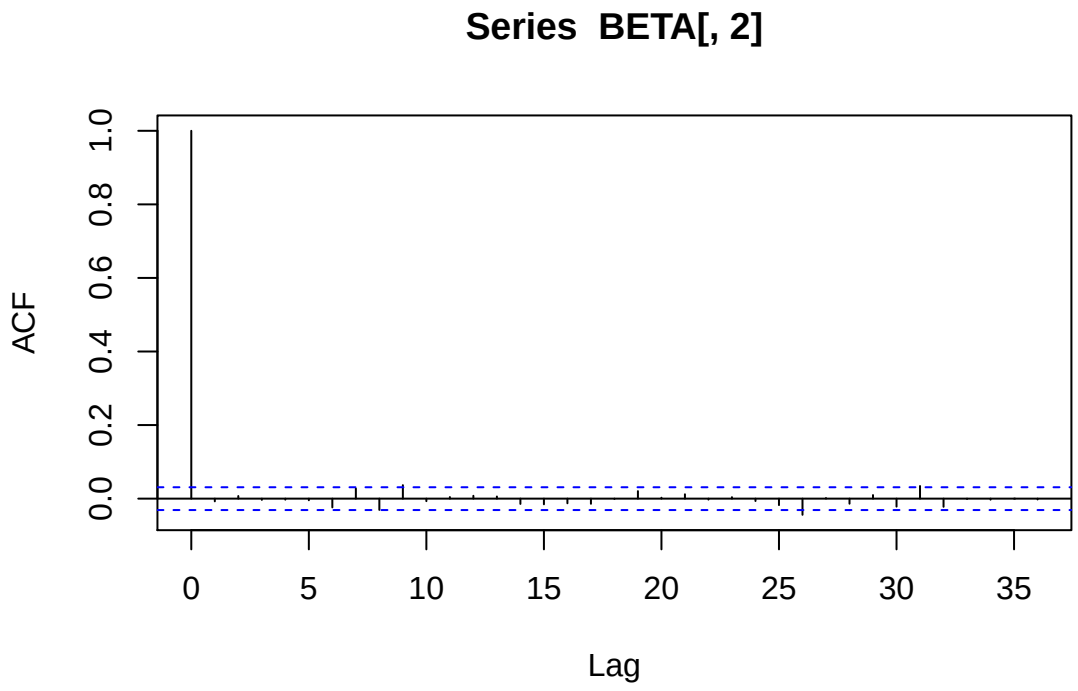
```r
apply(BETA, 2, p_not0)
```

```
## [1] 1.0000 0.9205 1.0000 1.0000
```

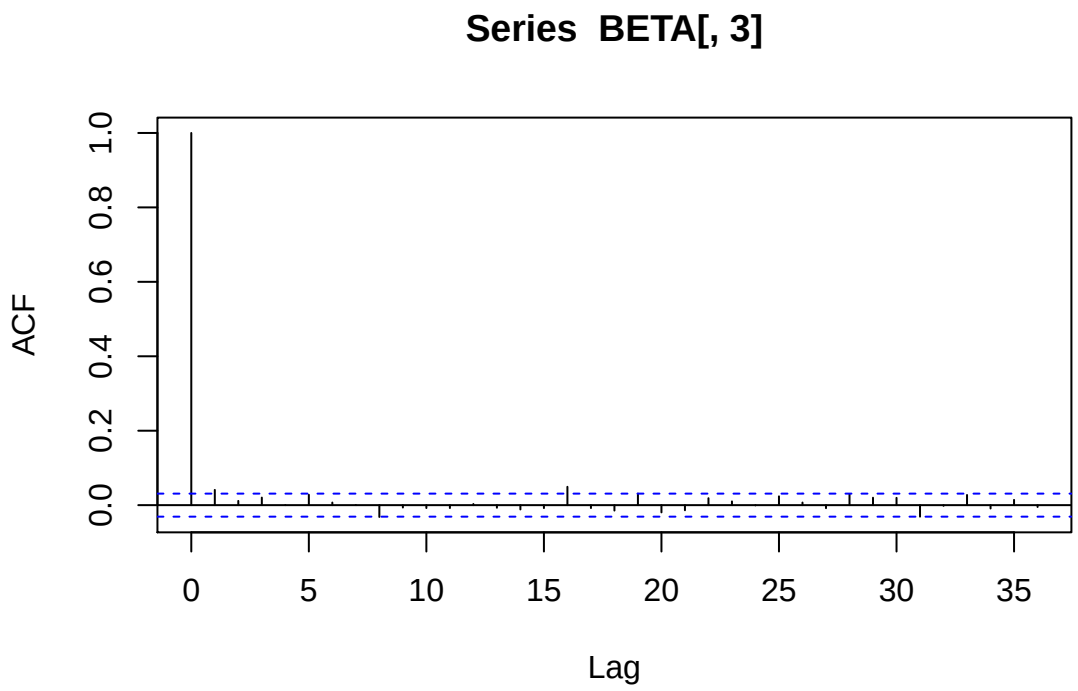
```r
coda::geweke.diag(beta[-c(1:burnIn),])
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##    var1    var2    var3    var4
##  1.2485 -0.8277  1.3089 -1.4411
```

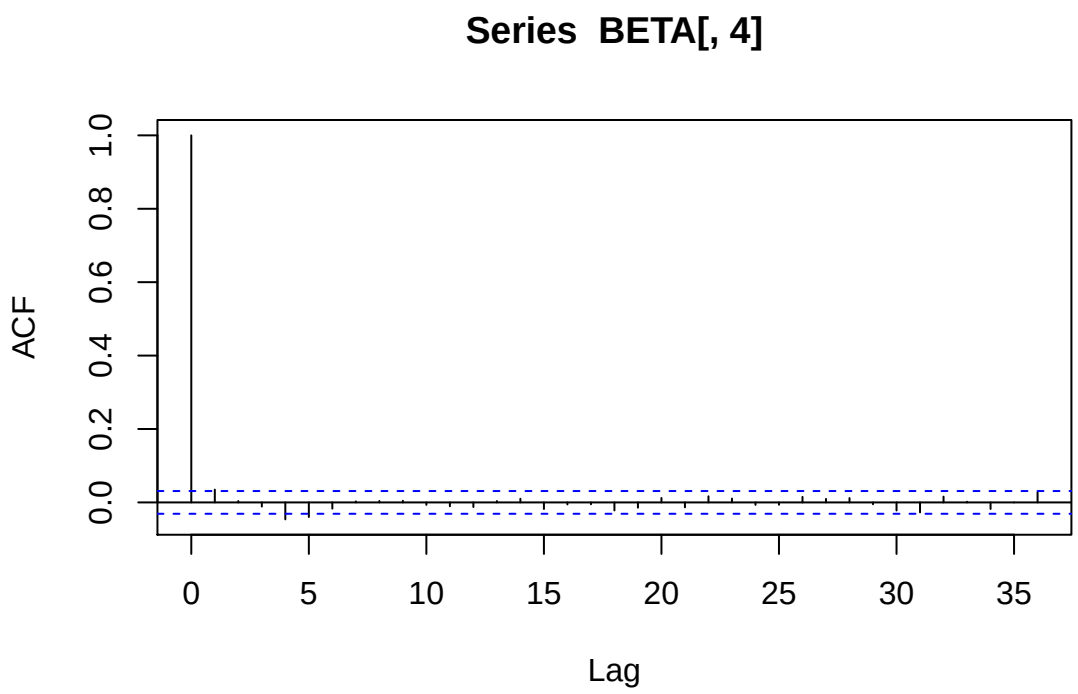
```r
acf(BETA[, 2])
```


Series BETA[, 2]
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
35
Lag

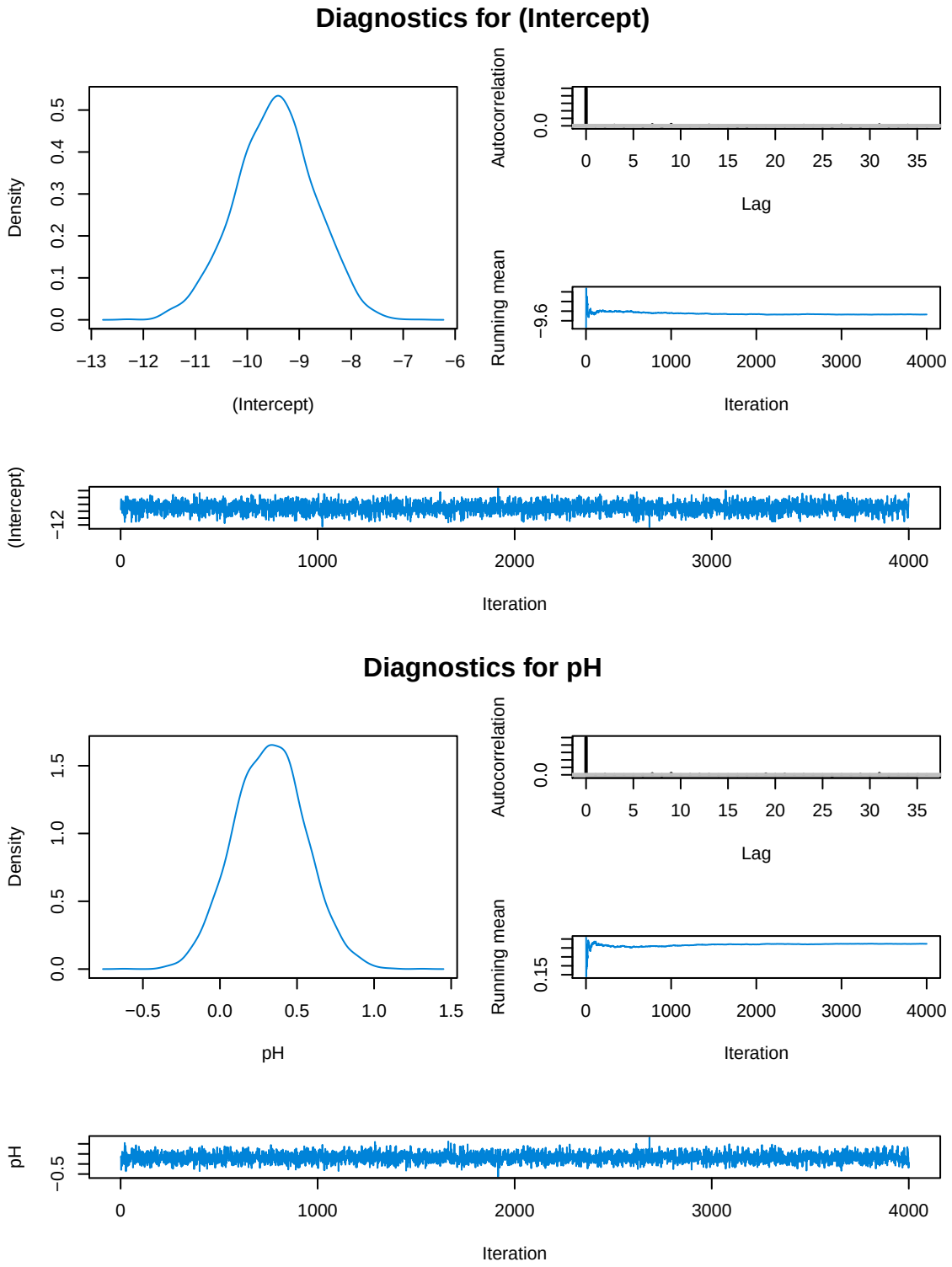
```
acf(BETA[, 3])
```

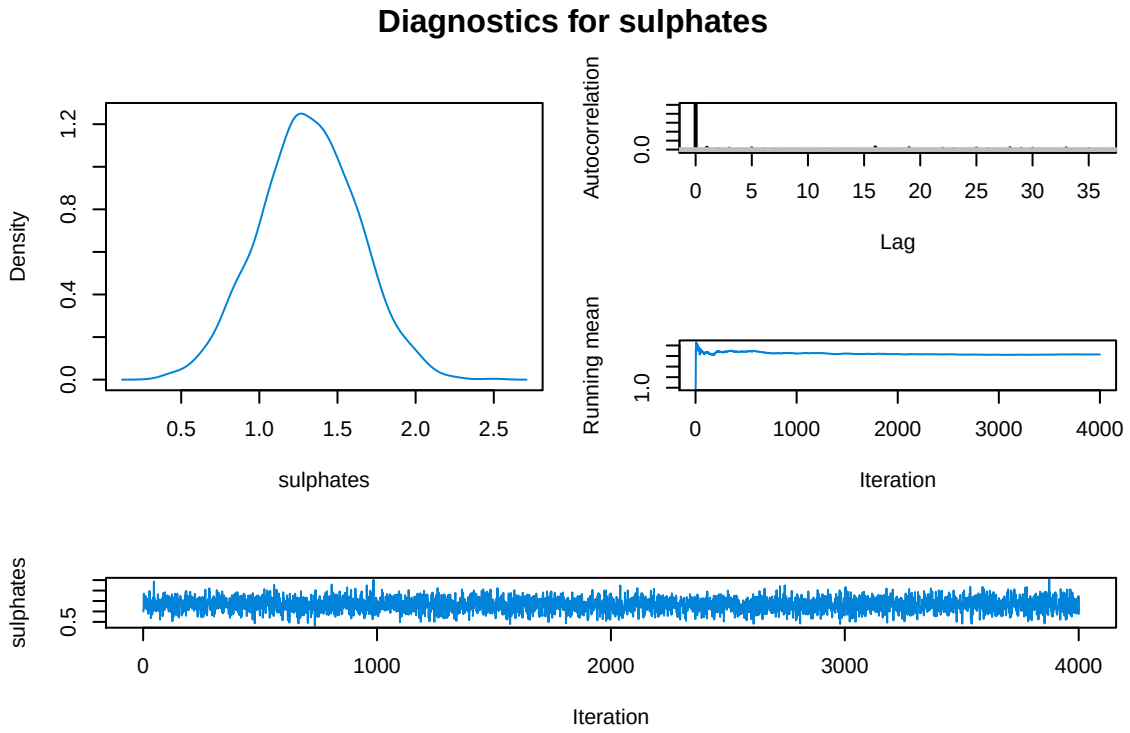

Series BETA[, 3]
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
35
Lag

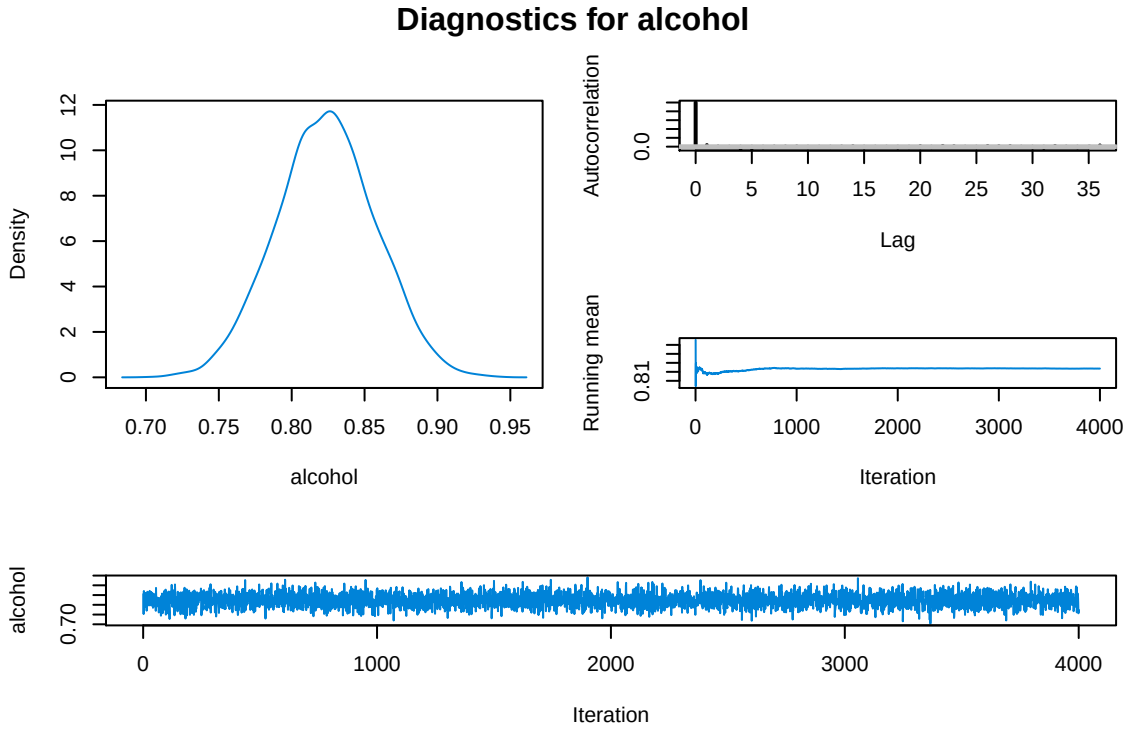
```r
acf(BETA[, 4])
```




```r
colnames(BETA) <- colnames(X)
for (v in colnames(BETA)) {
  x <- data.frame(var = BETA[,v])
  colnames(x) <- v
  mcmcplots::mcmcplot1(x, style = "plain")
}
```


Diagnostics for (Intercept)
Density
(Intercept)
Autocorrelation
Lag
Running mean
Iteration
(Intercept)
Iteration
Diagnostics for pH
Density
pH
Autocorrelation
Lag
Running mean
Iteration
pH
Iteration






```
lppdv       <- vector(length = n)
pwaicv      <- vector(length = n)
B <- 4000
for(i in 1:n){
```

```
  yi            <- Y[i]
  p_hat <- apply(BETA, 1, function(bb){
    exp(sum(X[i] * bb)) / (1 + exp(sum(X[i] * bb)))
  })
  liki      <- (p_hat^yi) * ( 1 - p_hat)^(1 - yi)
  lppdv[i]  <- 1/B*sum(liki)
  pwaicv[i] <- (1/(B-1))*sum((log(liki) - mean(log(liki)))^2)
}

lppd    <- sum(log(lppdv))
pwaic   <- (1/(B-1))*sum(pwaicv)
WAIC    <- -2*lppd +2*pwaic
WAIC
```

```
## [1] 44563.73
```

We find that the median estimate for each parameter is very close to that from the glm. We also find that the estimate for pH is not statistically significant, as 0 is in our 90% confidence interval and our probability that the magnitude is greater than 0 only comes out to about 80%.

Now use a multivariate normal prior, centered at (0,0,0,0), with $\Sigma = 9 * I_{4\times4}$:

$$\begin{aligned}\log P(\beta|x) &\propto \log((\exp\left[-1/2\beta^T\Sigma^{-1}\beta\right])\prod\hat{p}_i^{y_i}(1-\hat{p}_i)^{1-y_i}) \\ &= \log((\exp\left[-1/2\beta^T\Sigma^{-1}\beta\right]) + \sum\log\left(\hat{p}_i^{y_i}(1-\hat{p}_i)^{1-y_i}\right) \\ &= \left[-1/2\beta^T\Sigma^{-1}\beta\right] + \sum\log\left(\hat{p}_i^{y_i}(1-\hat{p}_i)^{1-y_i}\right)\end{aligned}$$

```
logL    <- function(b, x, y){
  p_hat <- exp(x %*% b) / (1 + exp(x %*% b))
  lL  <- sum(
    log(
      p_hat^y * (1 - p_hat)^(1 - y)
    )
  ) + (-1/2 * t(b) %*% solve(diag(9, 4)) %*% b)
  return(lL)
}

burnIn <- 10000
thin <- 25
B         <- burnIn + 4000*thin # burn-in + keep * thin

beta        <- matrix(0, nrow = B, ncol = ncol(X))
ar          <- vector('numeric', length = B)

beta[1,]    <- bhat # posterior mode

tau <- 1.5 # need to tune this

set.seed(1908)
for(t in 2:B){

  bstar <- mvtnorm::rmvnorm(1, beta[t-1,], tau*vbeta)
```

```
  r     <- exp(logL(t(bstar), X, Y)  -  logL(beta[t-1,], X, Y) )
  U     <- runif(1)

  if(U < min(1,r)){
    beta[t,]    <- bstar
    ar[t]         <- 1
  } else{
    beta[t,]    <- beta[t-1,]
    ar[t]         <- 0
  }

}

mean(ar[-c(1:burnIn)])
```

```
## [1] 0.28073
```

```
BETA <- beta[seq(burnIn + 1, B, thin), ]
coefs <- apply(BETA, 2, quantile, probs = c(0.5, 0.05, 0.95))
colnames(coefs) <- colnames(X)
round(t(coefs), 4)
```

```
##                50%       5%     95%
## (Intercept) -8.9051 -10.0701 -7.7108
## pH           0.1742  -0.1729  0.5307
## sulphates    1.2708   0.7598  1.7909
## alcohol      0.8153   0.7620  0.8698
```

```
coef(fit)
```

```
## (Intercept)          pH   sulphates     alcohol
##  -9.4447477   0.3175969   1.3055717   0.8232929
```

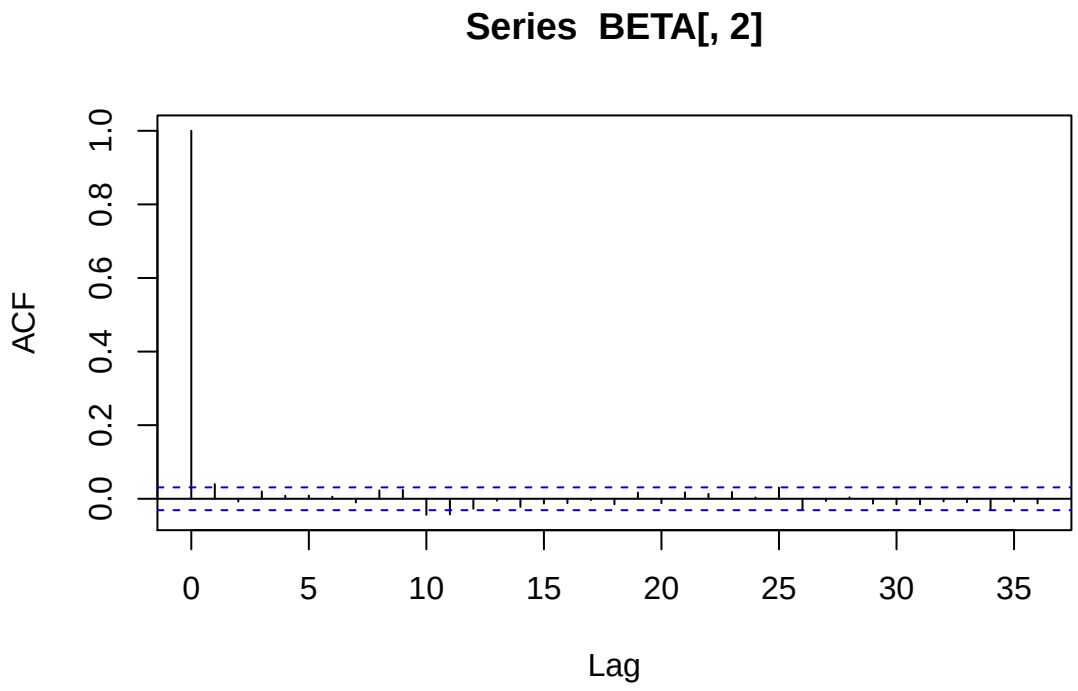
```
apply(BETA, 2, p_not0)
```

```
## [1] 1.000 0.789 1.000 1.000
```

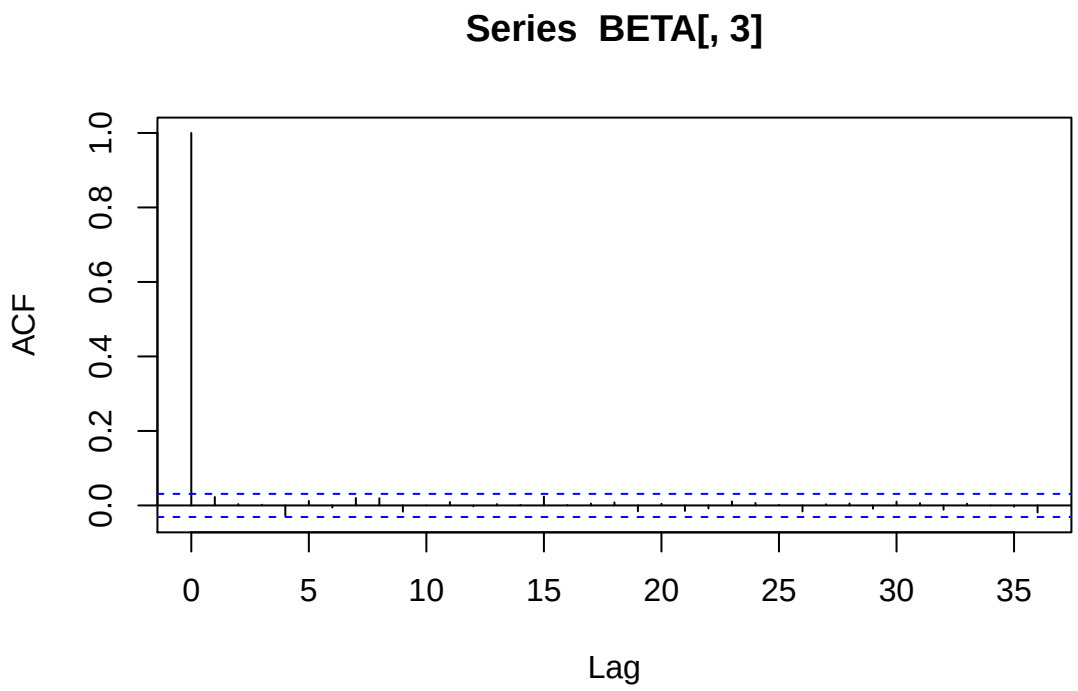
```
coda::geweke.diag(beta[-c(1:burnIn),])
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##    var1    var2    var3    var4
##  0.8407 -0.5220  0.8382 -0.9824
```

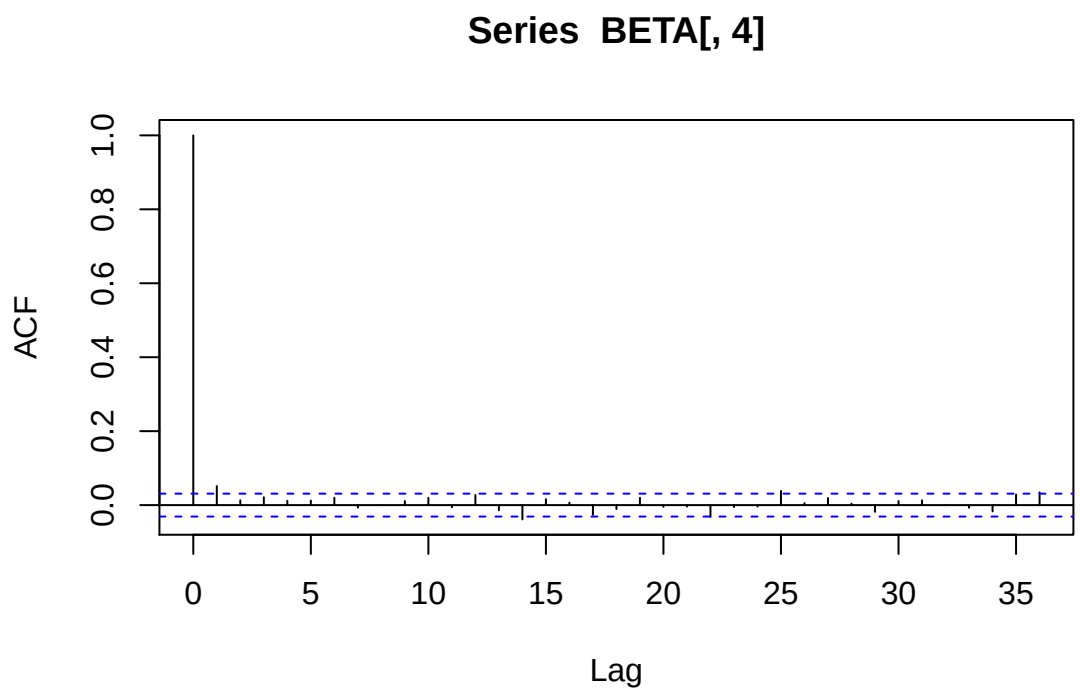
```
acf(BETA[, 2])
```


Series BETA[, 2]
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
35
Lag

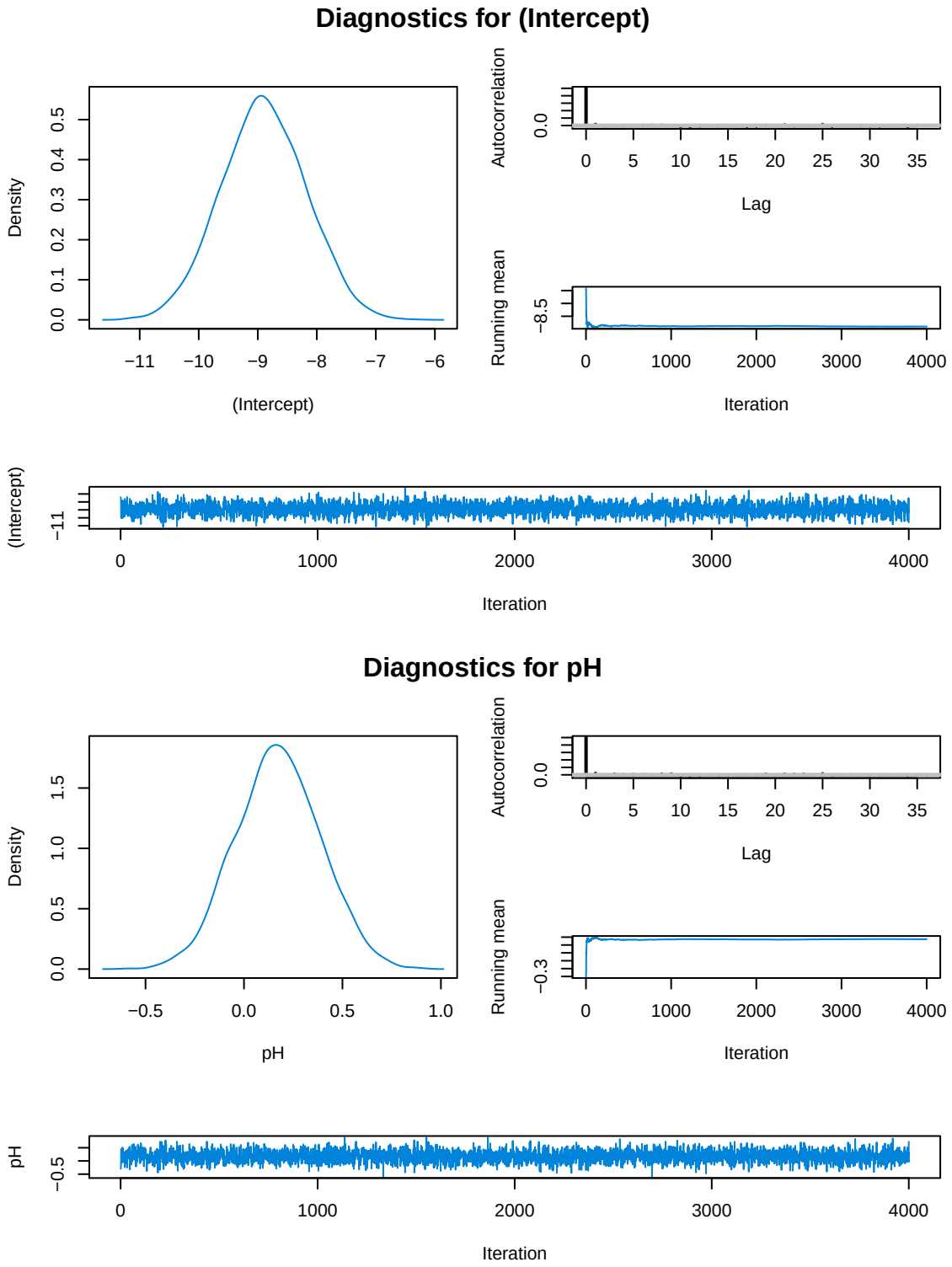
```
acf(BETA[, 3])
```

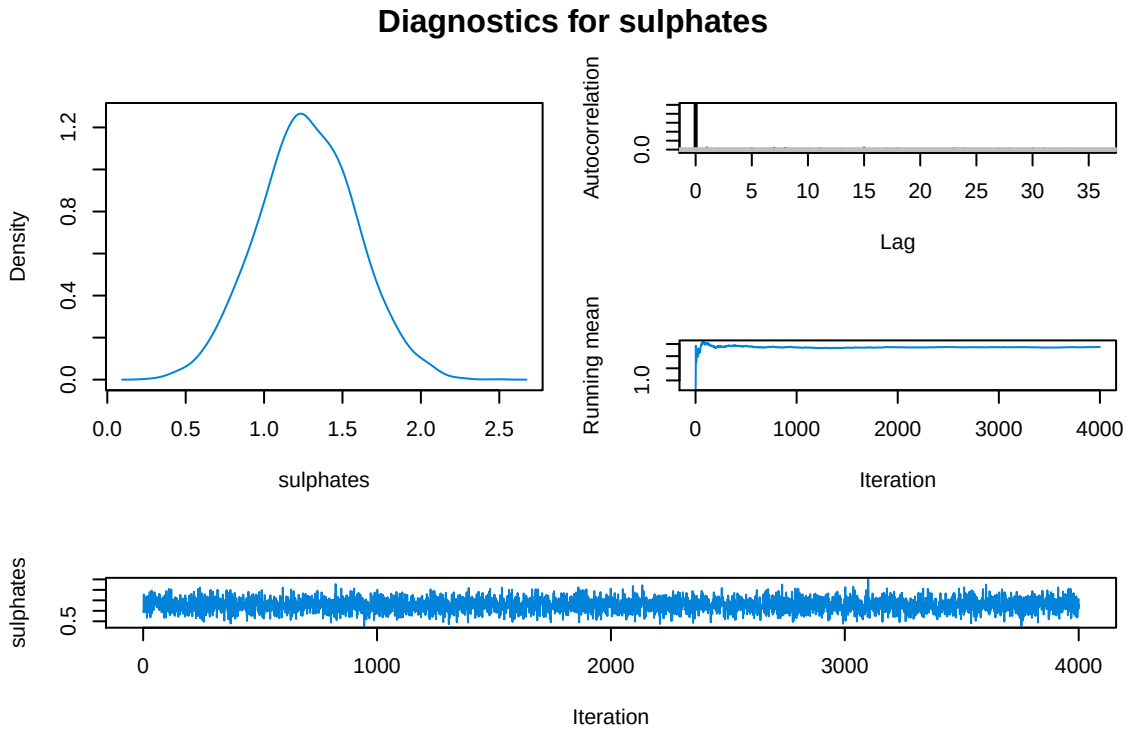

Series BETA[, 3]
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
35
Lag

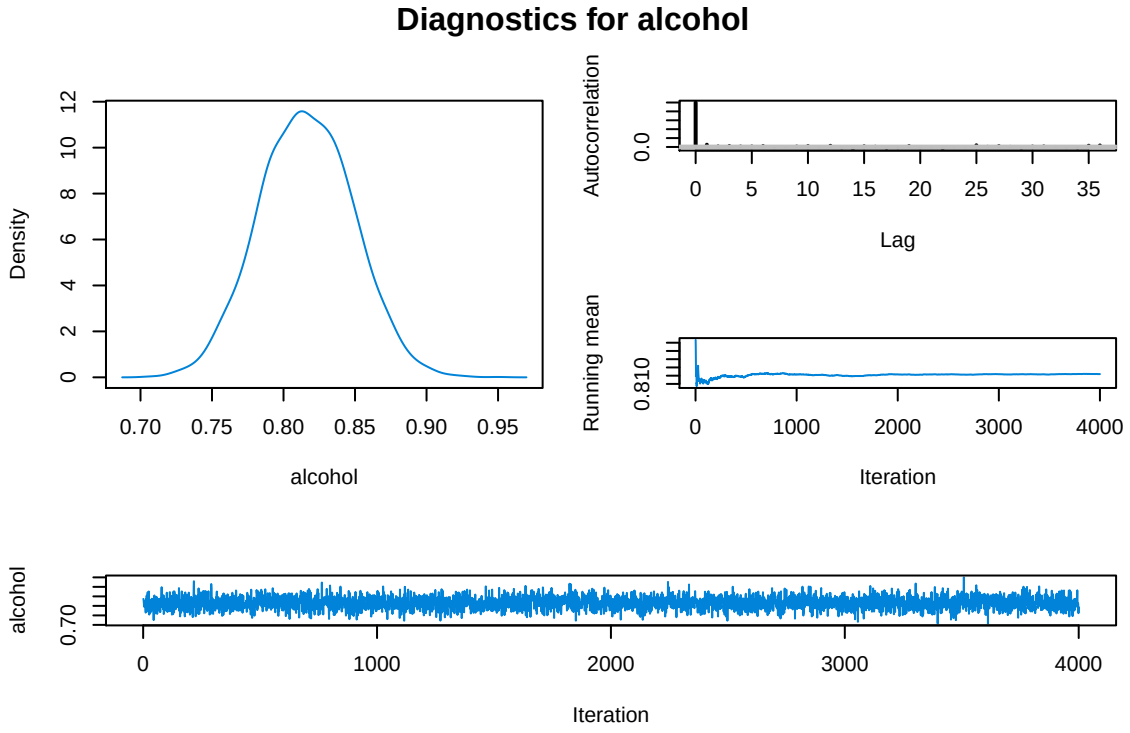
```r
acf(BETA[, 4])
```




```r
colnames(BETA) <- colnames(X)
for (v in colnames(BETA)) {
  x <- data.frame(var = BETA[,v])
  colnames(x) <- v
  mcmcplots::mcmcplot1(x, style = "plain")
}
```


Diagnostics for (Intercept)
Density
(Intercept)
Autocorrelation
Lag
Running mean
Iteration
(Intercept)
Iteration
Diagnostics for pH
Density
pH
Autocorrelation
Lag
Running mean
Iteration
pH
Iteration






```
lppdv       <- vector(length = n)
pwaicv      <- vector(length = n)
B <- 4000
for(i in 1:n){
```

```
  yi            <- Y[i]
  p_hat <- apply(BETA, 1, function(bb){
    exp(sum(X[i] * bb)) / (1 + exp(sum(X[i] * bb)))
  })
  liki      <- (p_hat^yi) * ( 1 - p_hat)^(1 - yi)
  lppdv[i]  <- 1/B*sum(liki)
  pwaicv[i] <- (1/(B-1))*sum((log(liki) - mean(log(liki)))^2)
}

lppd    <- sum(log(lppdv))
pwaic   <- (1/(B-1))*sum(pwaicv)
WAIC    <- -2*lppd +2*pwaic
WAIC
```

```
## [1] 42205.99
```

Now we find that we get slightly different coefficients from the glm when we have a prior that the coefficients are centered at 0. Particularly, the *pH* coefficient and the intercept are the most different, while the *sulphates* and *alcohol* coefficients are pretty close to our previous values. Also, we again find that the *pH* is not statistically significant, as 0 is in our confidenct interval, and we only see about a 78% chance that it is not 0.

We also get a lower WAIC with a prior centered at 0, which is indicative of a better overall model. Thus, having a prior with non-predictive coefficients was better for this data.