STA360 Homework 6 (Ken Ye)

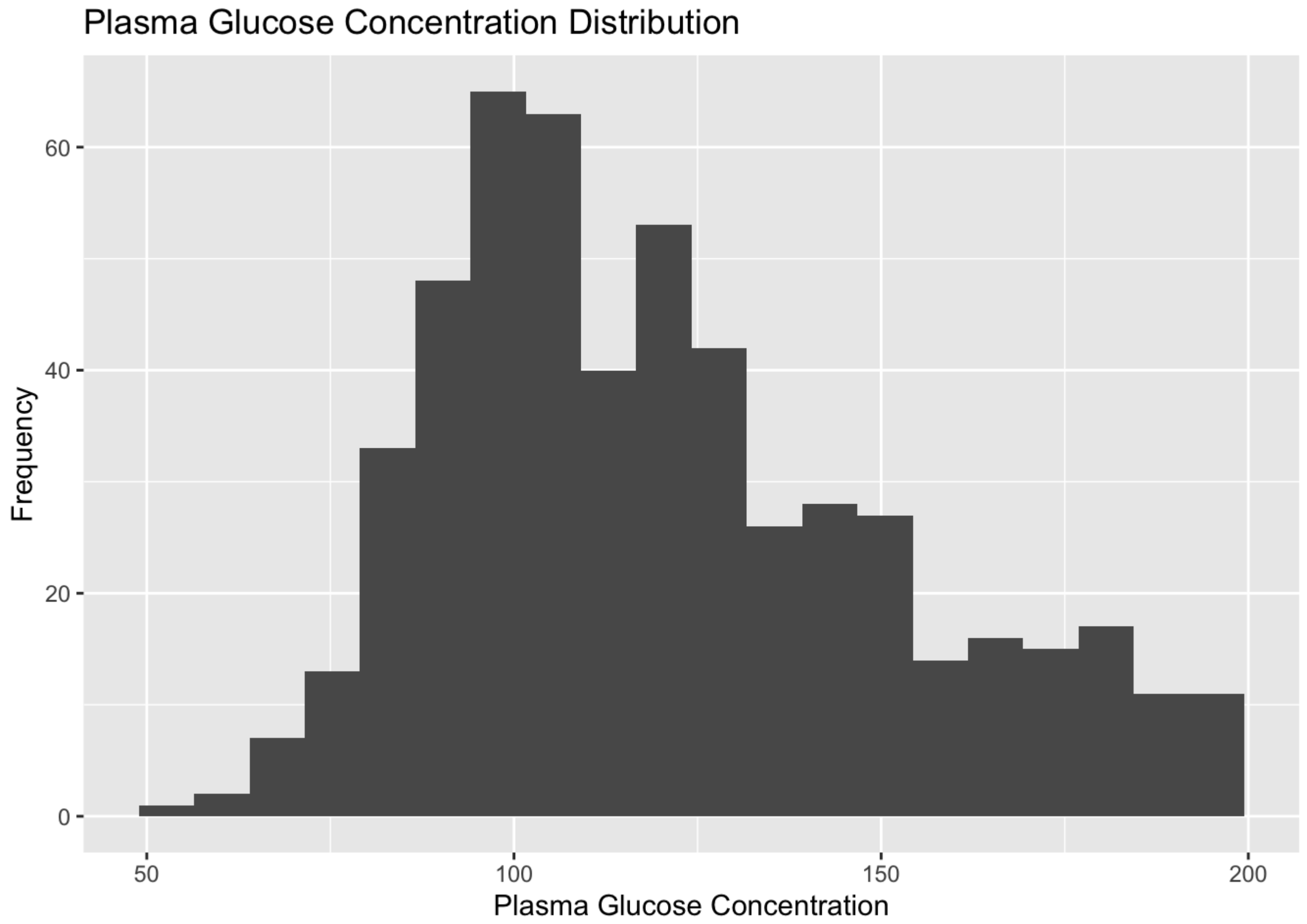
```
library(latex2exp)
library(coda)
library(ggplot2)
set.seed(0)
```

Exercise 6.2

```
# load data
glucose <- read.table("glucose.dat.txt")
```

Part a

```
# plot historgram
glucose |>
  ggplot(aes(x = V1)) +
  geom_histogram(bins = 20) +
  labs(title = "Plasma Glucose Concentration Distribution") +
  xlab("Plasma Glucose Concentration") +
  ylab("Frequency")
```



This empirical plasma glucose concentration distribution is not normal, as it”“s bimodal and significantly right-skewed.

Part b

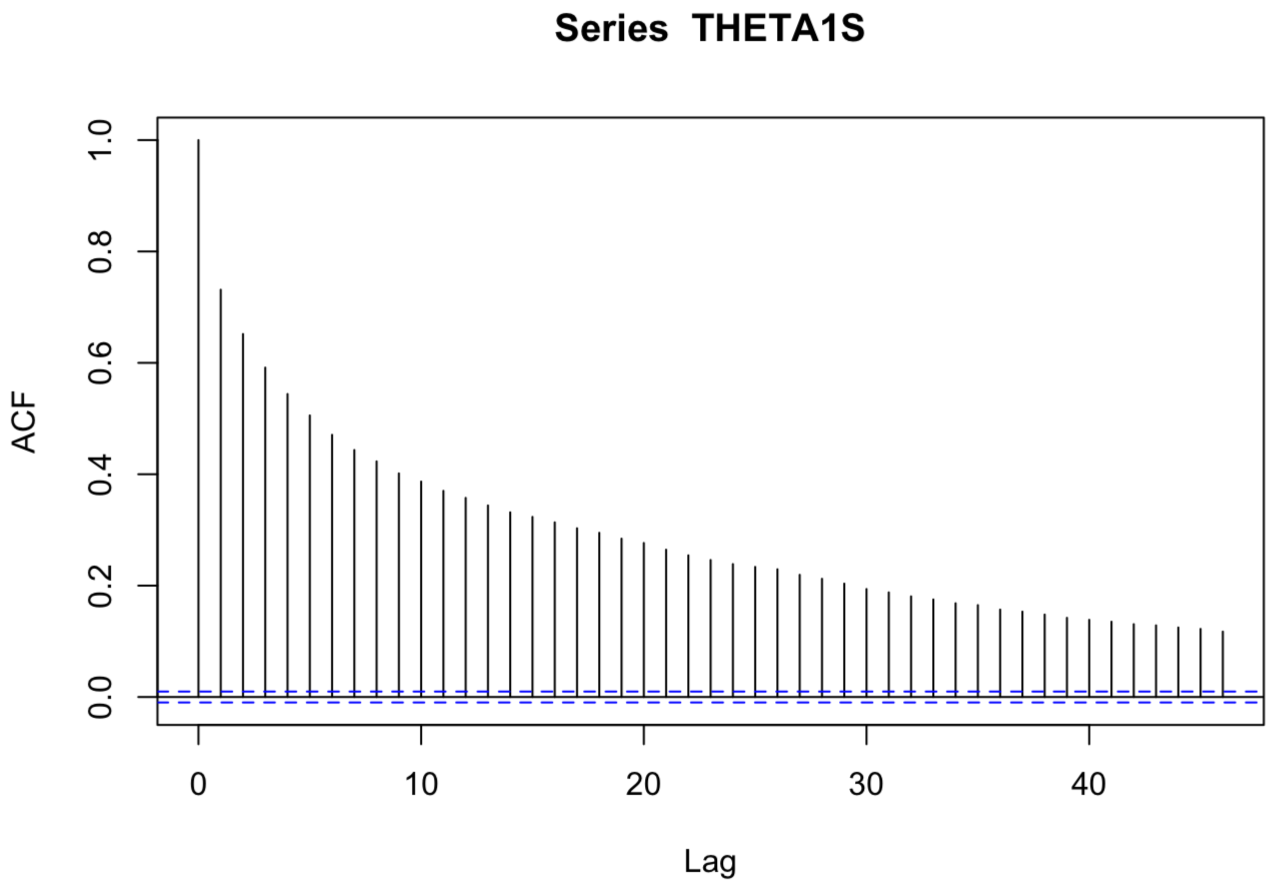
(See hand-written page)

Part c

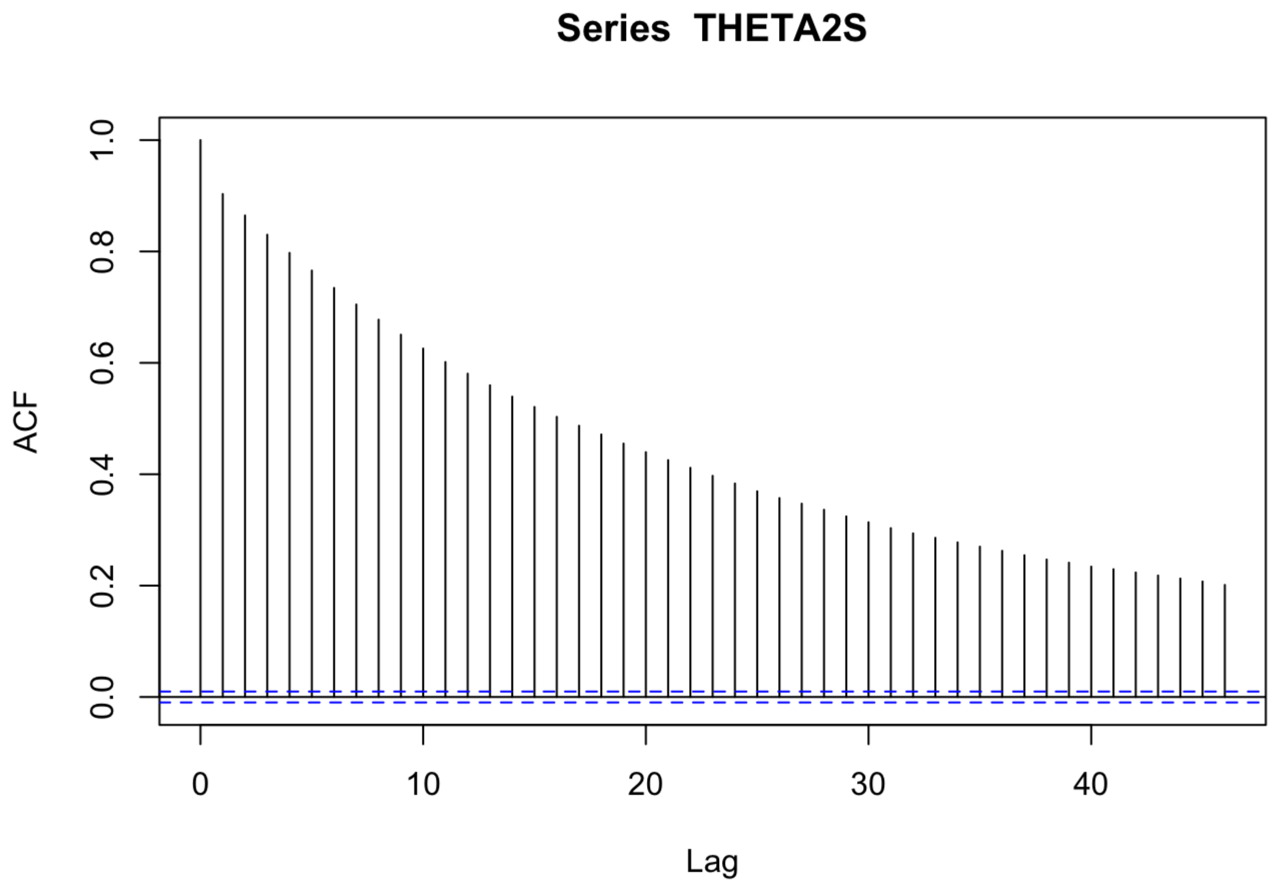
```r
Y <- glucose$V1
n <- length(Y)
nsim <- 5e4
burnin <- 1e4

# priors
a <- 1
b <- 1
mu0 <- 120
t20 <- 200
s20 <- 1000
nu0 <- 10

# storage vectors
THETA1 <- numeric(nsim - burnin)
THETA2 <- numeric(nsim - burnin)
Y.gibb <- numeric(nsim - burnin)

# starting values
p <- 0.5
theta1 <- mean(Y)
theta2 <- mean(Y)
s21 <- var(Y)
s22 <- var(Y)

# Gibbs sampling
for (s in 1:nsim) {
  # simulate X
  p.x1 <- p * dnorm(Y, theta1, sqrt(s21))
  p.x2 <- (1 - p) * dnorm(Y, theta2, sqrt(s22))
  p.xi <- p.x1 / (p.x1 + p.x2)
  X <- rbinom(n, 1, p.xi)

  # calcuate group-specific summary statistics
  n1 <- sum(X)
  n2 <- n - n1
```

```
y1 <- Y[X == 1]
y2 <- Y[X == 0]
ybar1 <- mean(y1)
ybar2 <- mean(y2)
yvar1 <- var(y1)
yvar2 <- var(y2)

# simulate p
p <- rbeta(1, a + n1, b + n2)

# simulate theta1
t2n1 <- 1 / (1 / t20 + n1 / s21)
mun1 <- (mu0 / t20 + n1 * ybar1 / s21) / (1 / t20 + n1 / s21)
theta1 <- rnorm(1, mun1, sqrt(t2n1))

# simulate theta2
t2n2 <- 1 / (1 / t20 + n2 / s22)
mun2 <- (mu0 / t20 + n2 * ybar2 / s22) / (1 / t20 + n2 / s22)
theta2 <- rnorm(1, mun2, sqrt(t2n2))

# simulate sigma^2 1
nun1 <- nu0 + n1
s2n1 <- (nu0 * s20 + (n1 - 1) * yvar1 + n1 * (ybar1 - theta1)^2) / nun1
s21 <- 1 / rgamma(1, nun1 / 2, s2n1 * nun1 / 2)

# simulate sigma^2 2
nun2 <- nu0 + n2
s2n2 <- (nu0 * s20 + (n2 - 1) * yvar2 + n2 * (ybar2 - theta2)^2) / nun2
s22 <- 1 / rgamma(1, nun2 / 2, s2n2 * nun2 / 2)

# simulate posterior
x.gibb <- runif(1) < p
y.gibb <- ifelse(x.gibb,
                rnorm(1, theta1, sqrt(s21)),
                rnorm(1, theta2, sqrt(s22)))

# store values
if (s > burnin){
    THETA1[s - burnin] <- theta1
    THETA2[s - burnin] <- theta2
    Y.gibb[s - burnin] <- y.gibb
  }
}
```

```
THETA1S <- pmin(THETA1, THETA2)
THETA2S <- pmax(THETA1, THETA2)

# plot autocorrelation
acf(THETA1S)
```



```
acf(THETA2S)
```



```
# print effective size
print("The effective size of theta (1) (s) is: ")
```

```
## [1] "The effective size of theta (1) (s) is: "
```

```
print(effectiveSize(THETA1S))
```

```
##     var1
## 1280.576
```

```
print("The effective size of theta (2) (s) is: ")
```

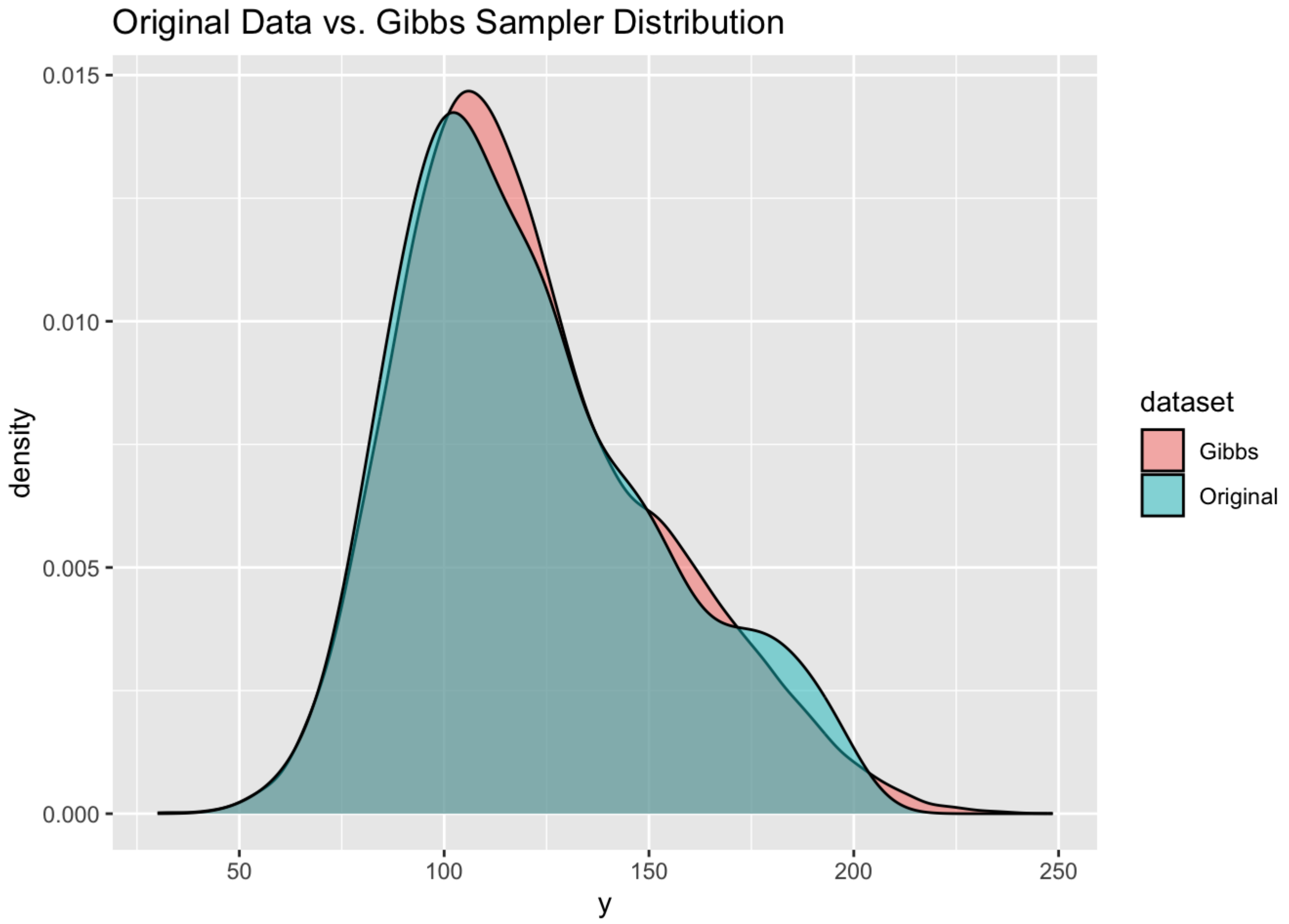
```
## [1] "The effective size of theta (2) (s) is: "
```

```
print(effectiveSize(THETA2S))
```

```
##     var1
## 875.6929
```

Part d

```
#
Y.compare = rbind(data.frame(y = Y.gibb, dataset = "Gibbs"),
              data.frame(y = Y, dataset = "Original"))
# plot
ggplot(Y.compare, aes(x = y, fill = dataset)) +
  geom_density(alpha = 0.5) +
  labs(title = "Original Data vs. Gibbs Sampler Distribution")
```



Based on the graph, the densities of the original and the Gibbs sampler are very close, indicating that this two-component mixture model is a good fit for the glucose data.

Exercise 6.3

```
# load data
divorce <- read.table("divorce.dat.txt")
```

```
# function for simulating from a constrained normal distribution with mean mean and standard deviation sd, constr
ained to lie in the interval (a,b)
rcnorm <- function(n, mean = 0, sd = 1, a = -Inf, b = Inf){
  u <- runif(n, pnorm((a - mean) / sd), pnorm((b - mean) / sd) )
  mean + sd * qnorm(u)
}
```

Part a

(See hand-written page)

Part b

(See hand-written page)

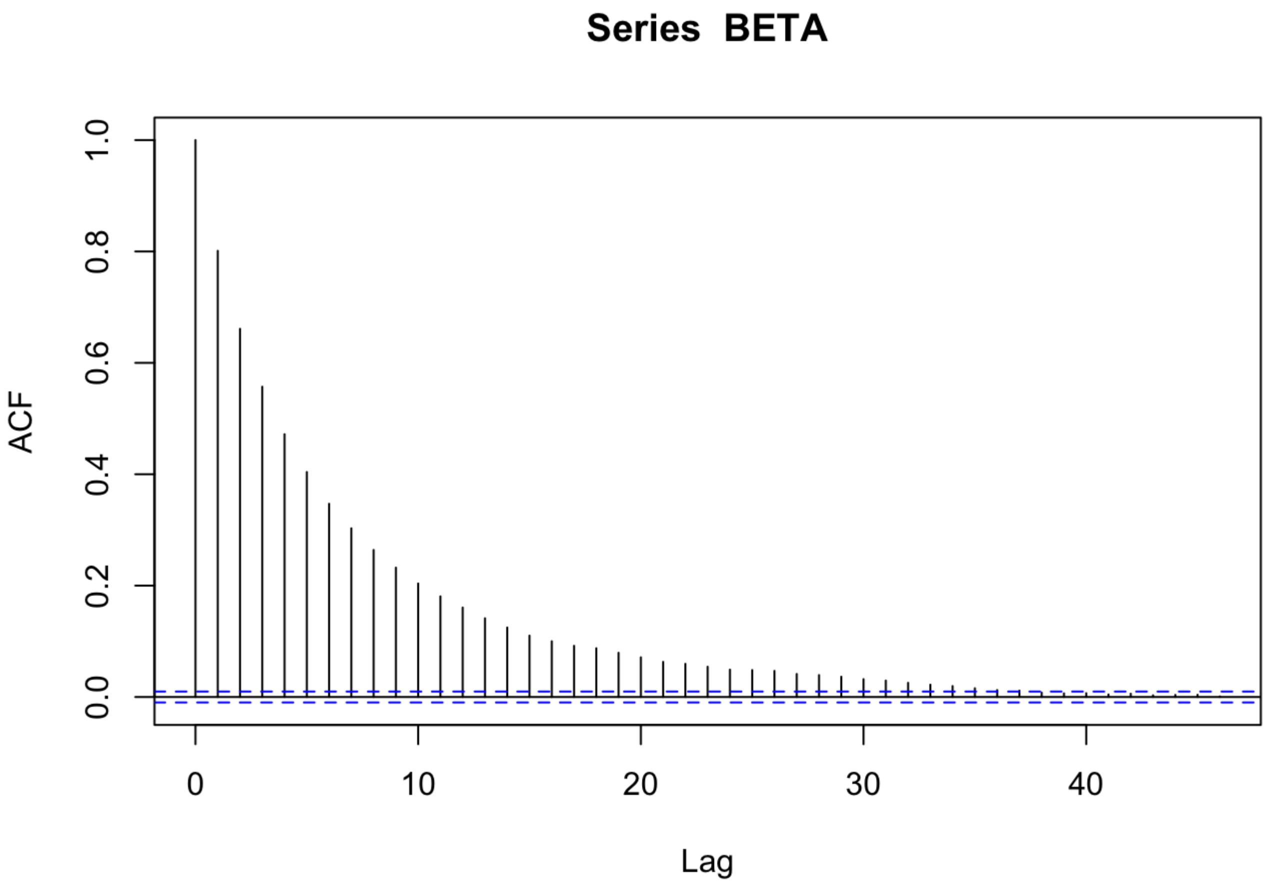
Part c

```r
Y <- divorce$V1
X <- divorce$V2
n <- length(Y)
nsim <- 5e4
burnin <- 1e4

# priors
t2b <- 16
t2c <- 16

# storage vectors
Z <- rep(list(0 * length(X)), times = nsim - burnin)
BETA <- numeric(nsim - burnin)
C <- numeric(nsim - burnin)

# starting values
z <- rep(0, n)
beta <- 0
c <- 0

# Gibbs sampling
for (s in 1:nsim) {
  # simulate beta
  mubn <- sum(z*X) / (sum(X^2) + 1/ (t2b))
  t2bn <- 1 / (sum(X^2) - 1 / t2b)
  beta <- rnorm(1, mubn, sqrt(t2bn))

  # simulate c
  z0 <- subset(z, Y == 0)
  z1 <- subset(z, Y == 1)
  a <- max(z0)
  b <- min(z1)
  c <- rcnorm(1, 0, sqrt(t2c), a, b)

  # simulate z
  for (i in 1:n){
    if(Y[i] == 1){
      z[i] <- rcnorm(1, beta * X[i], 1, c, Inf)
    }
    else{
      z[i] <- rcnorm(1, beta * X[i], 1, -Inf, c)
    }
  }

  # store values
  if (s > burnin){
    Z[[s - burnin]] <- z
    BETA[s - burnin] <- beta
    C[s - burnin] <- c
  }
}
```

```
# beta effective size and autocorrelation
effectiveSize(BETA)
```

```
##     var1
## 3338.234
```

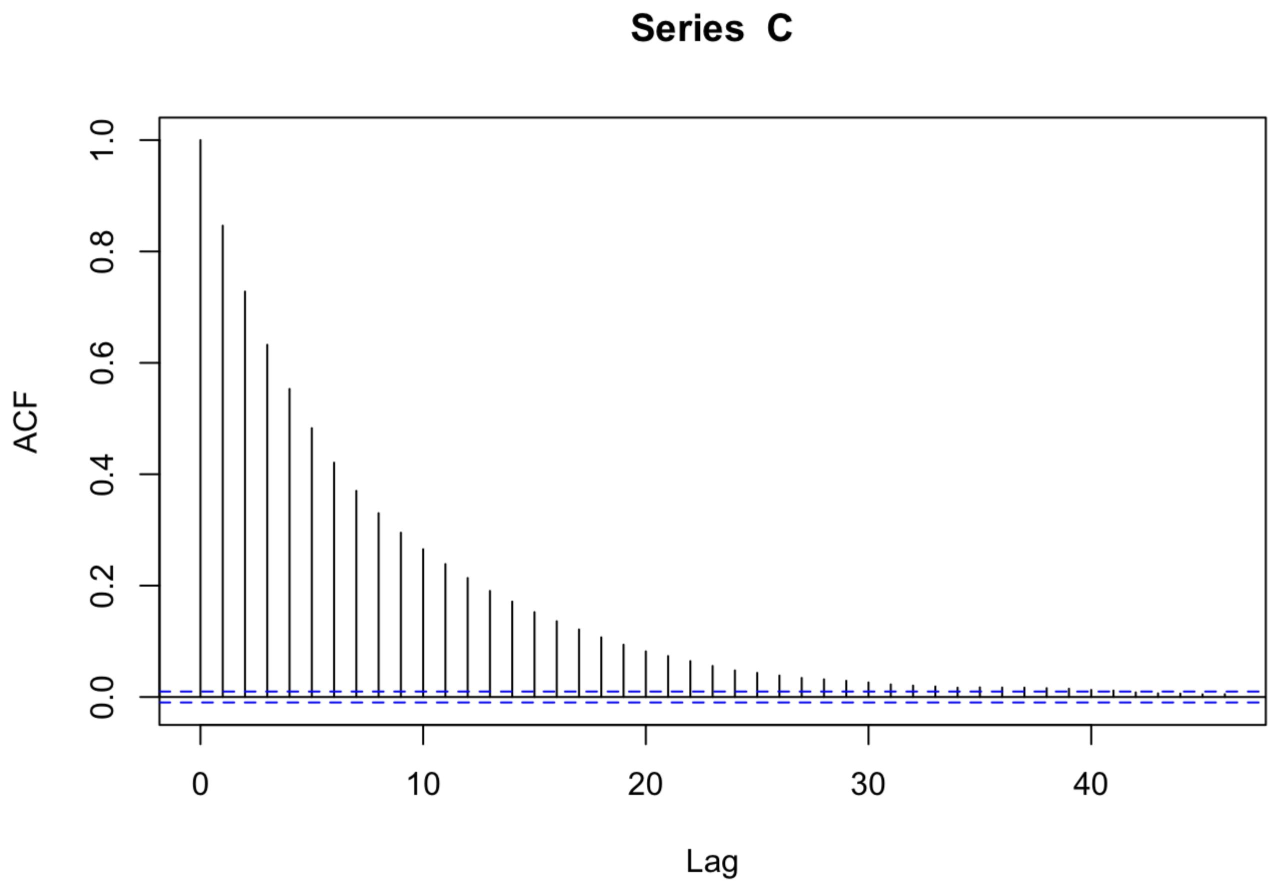
```
acf(BETA)
```



```
# c effective size and autocorrelation
effectiveSize(C)
```

```
##     var1
## 2611.212
```

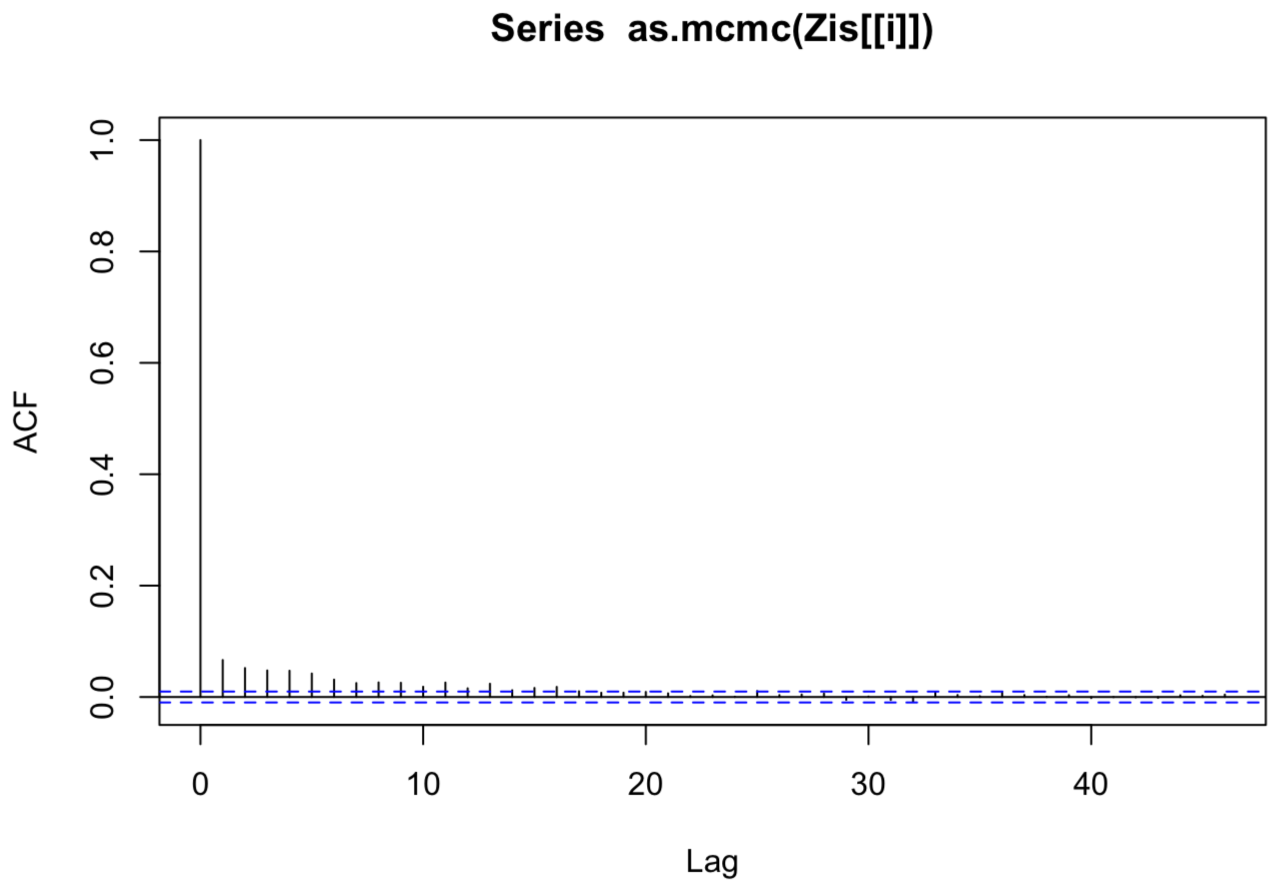
```
acf(C)
```

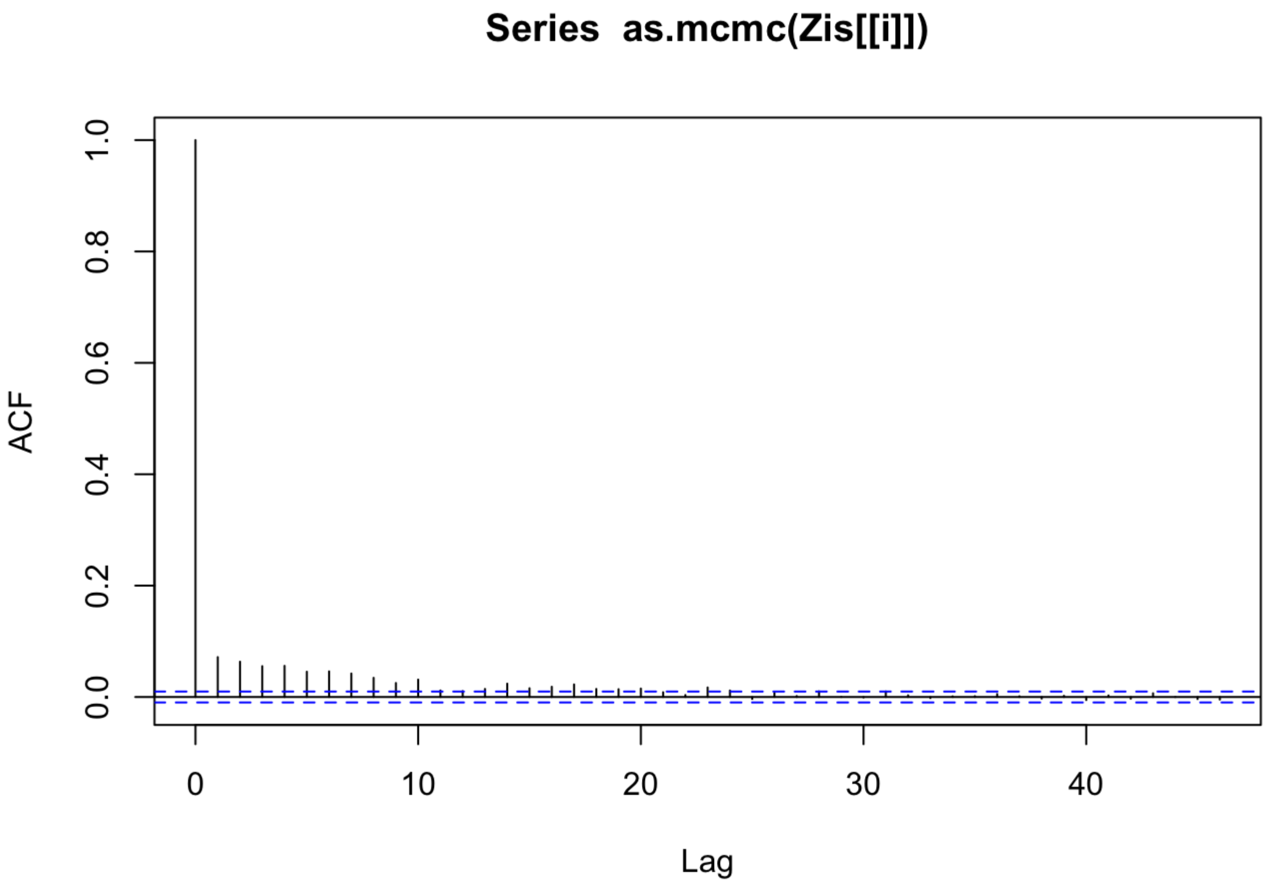

Series C
ACF
Lag
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40

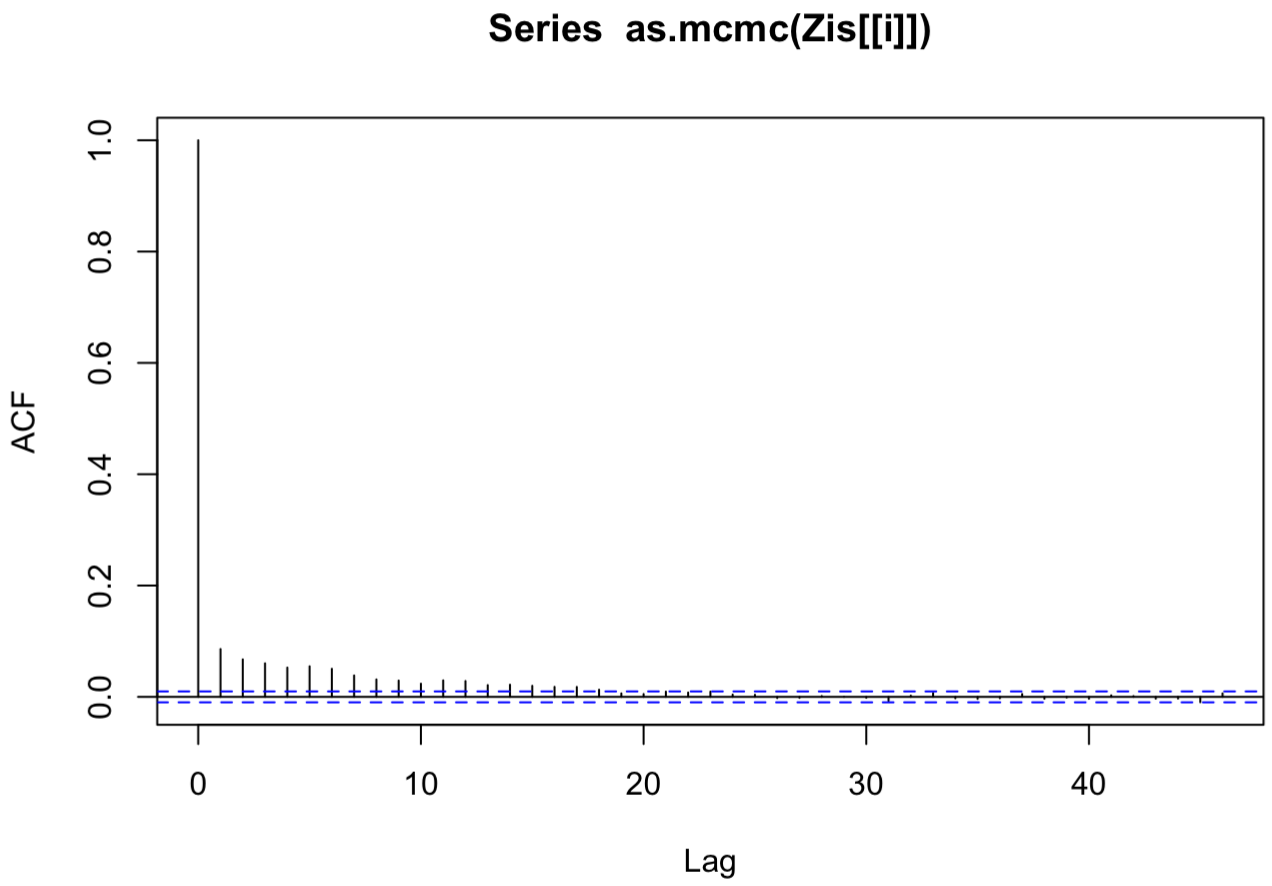
```r
# zi's effective size
Zis <- rep(list(0 * length(Z)), times = n)
for (i in 1:length(Z)){
  for (j in 1:n){
     Zis[[j]][i] <- Z[[i]][j]
  }
}
Zis.eff <- rep(0, times = length(Zis))
for (i in 1:length(Zis)){
  Zis.eff[i] <- effectiveSize(unlist(Zis[[i]]))
}
Zis.eff
```

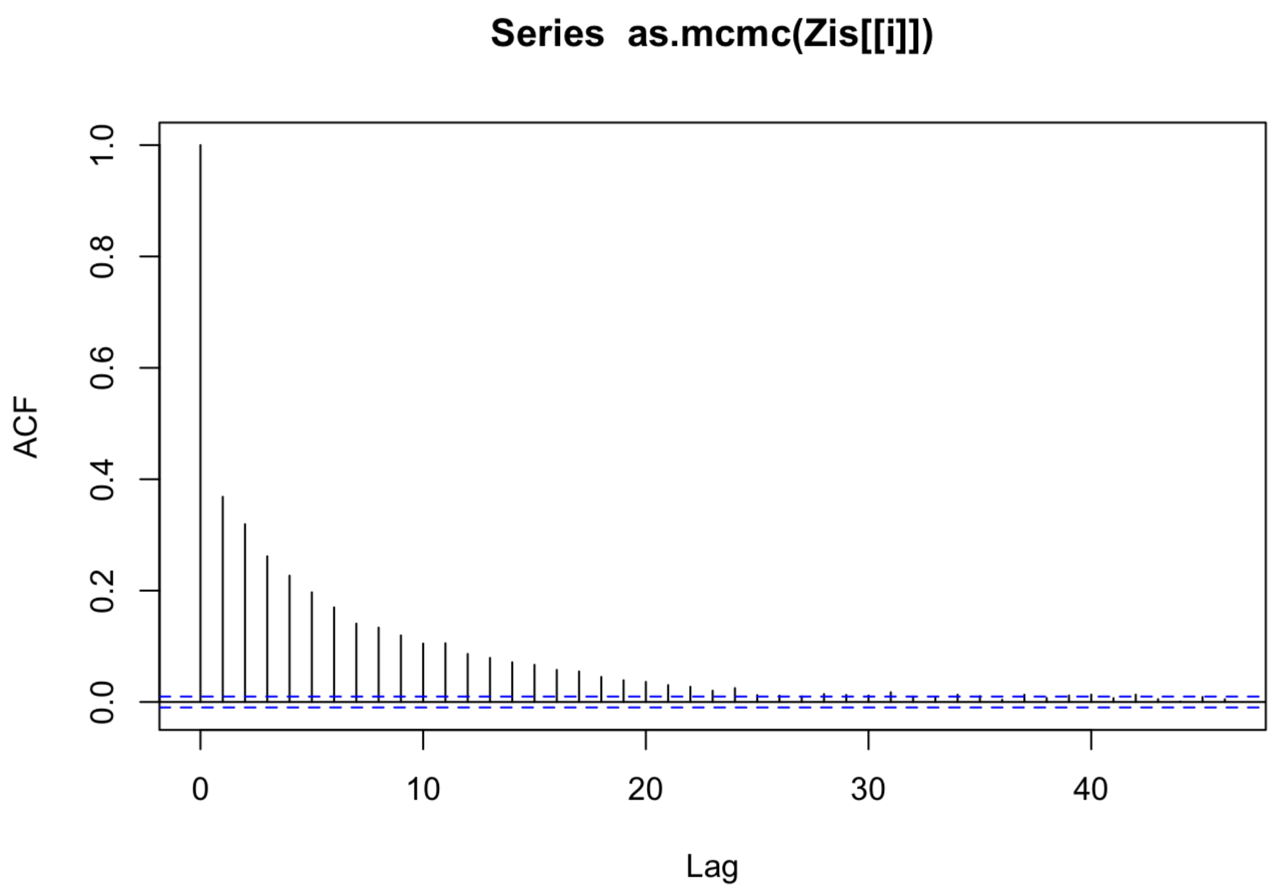
```
##  [1] 19027.034 17233.683 16214.720  5989.295 17960.810 19064.730 18205.878
##  [8]  6995.131 18500.697  6292.201 19794.650  3498.864  6996.895  7192.662
## [15]  6925.164 19898.214  6836.595  6516.958 17812.840 18809.866  6719.969
## [22] 19622.109 19797.833  6620.454 19957.361
```

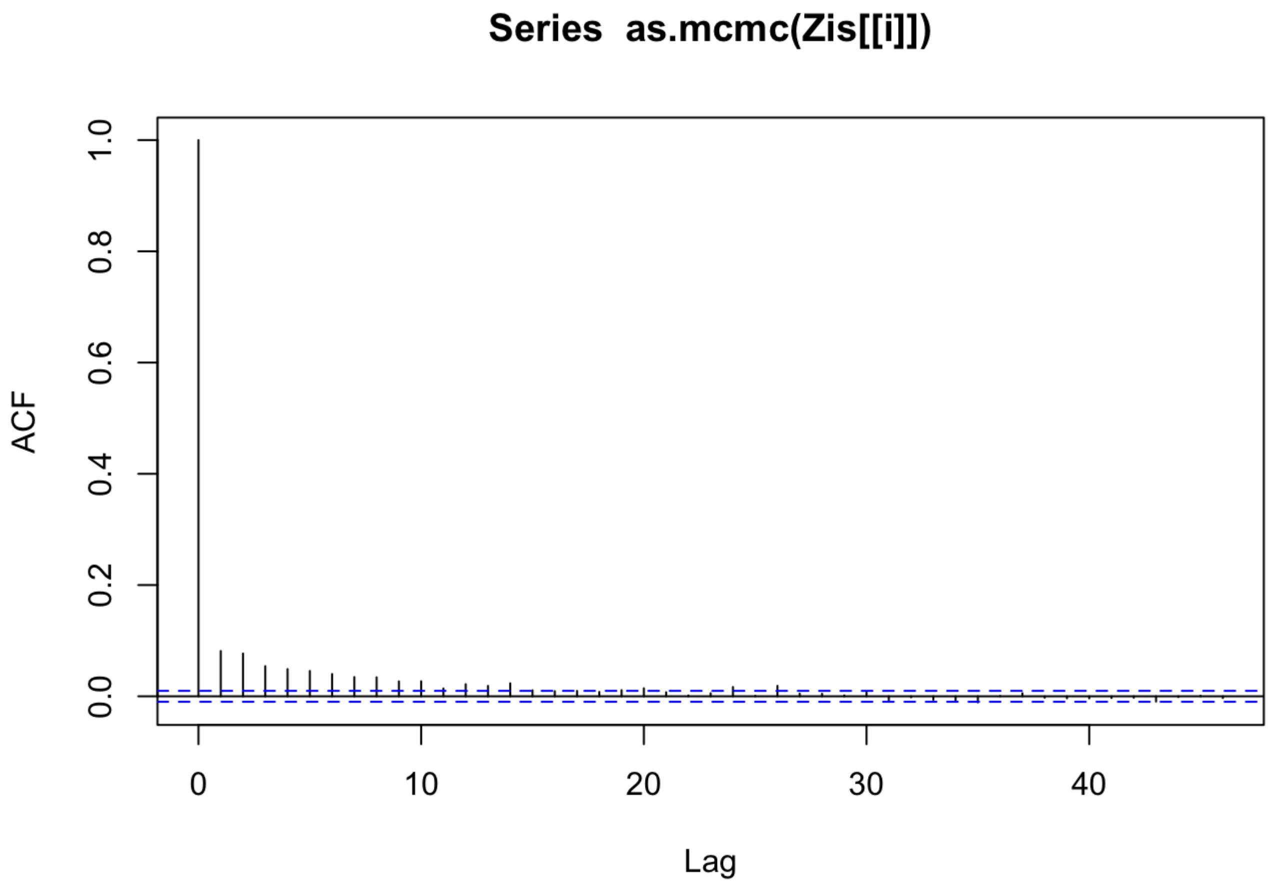
```r
# zi's autocorrelation
for (i in 1:length(Zis)){
  acf(as.mcmc(Zis[[i]]))
}
```

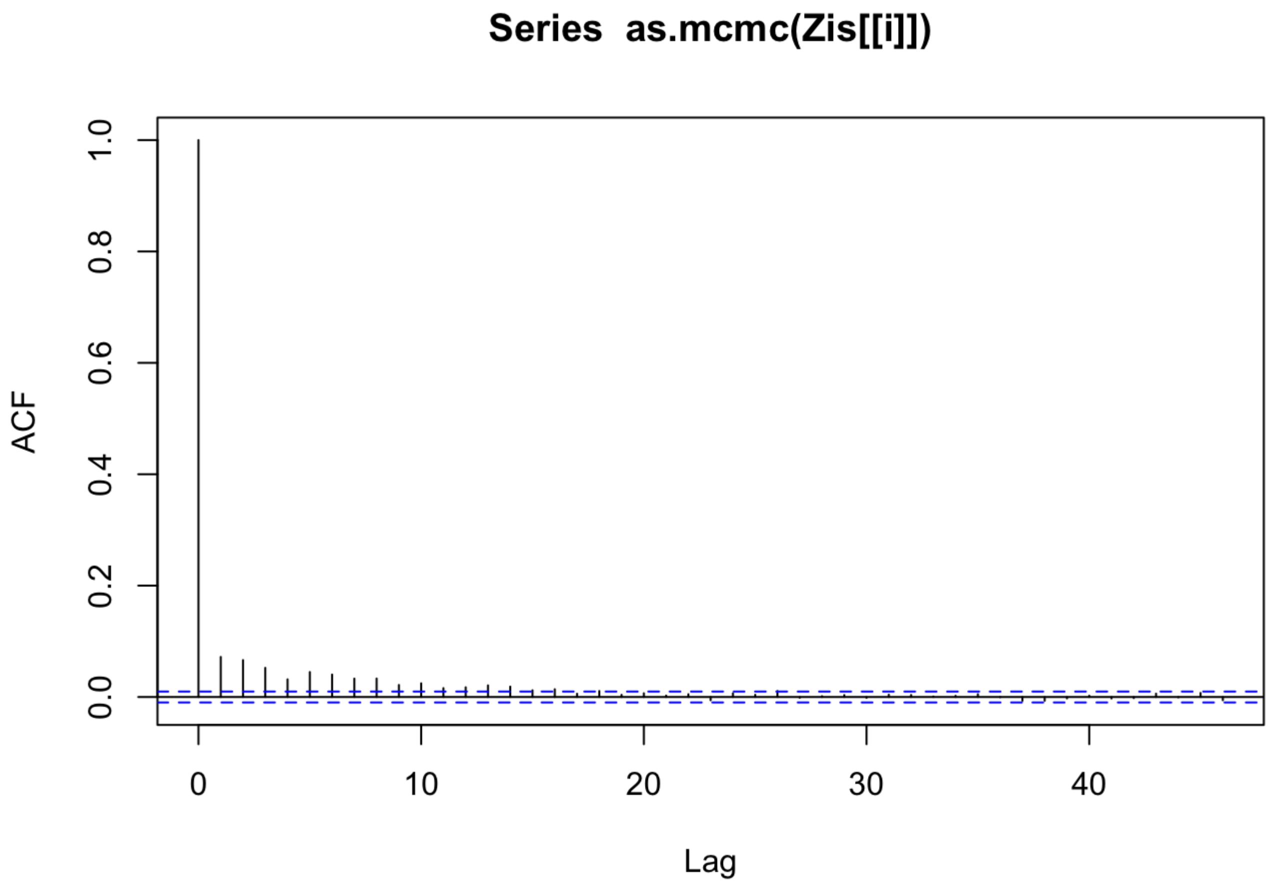

Series as.mcmc(Zis[[i]])
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
Lag

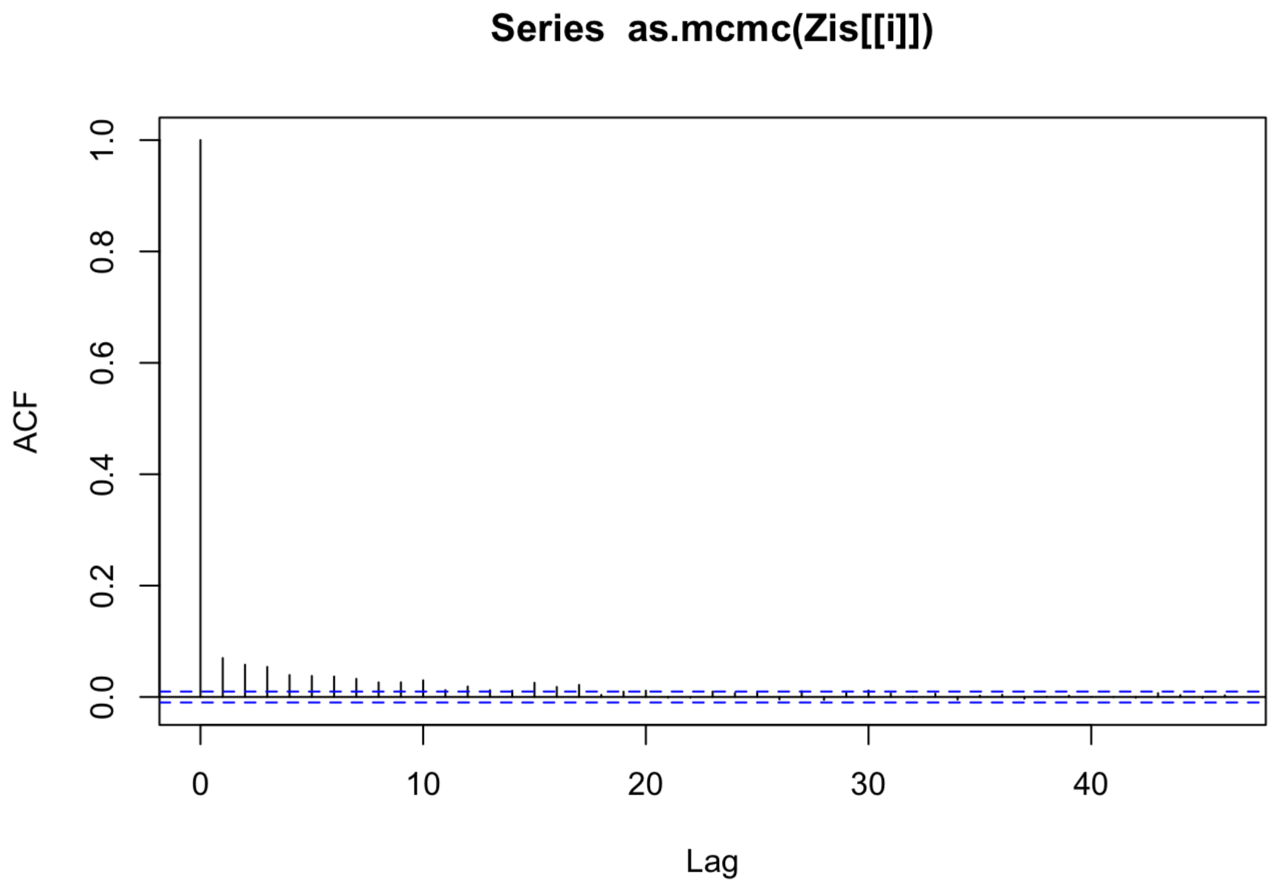

Series as.mcmc(Zis[[i]])
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
Lag

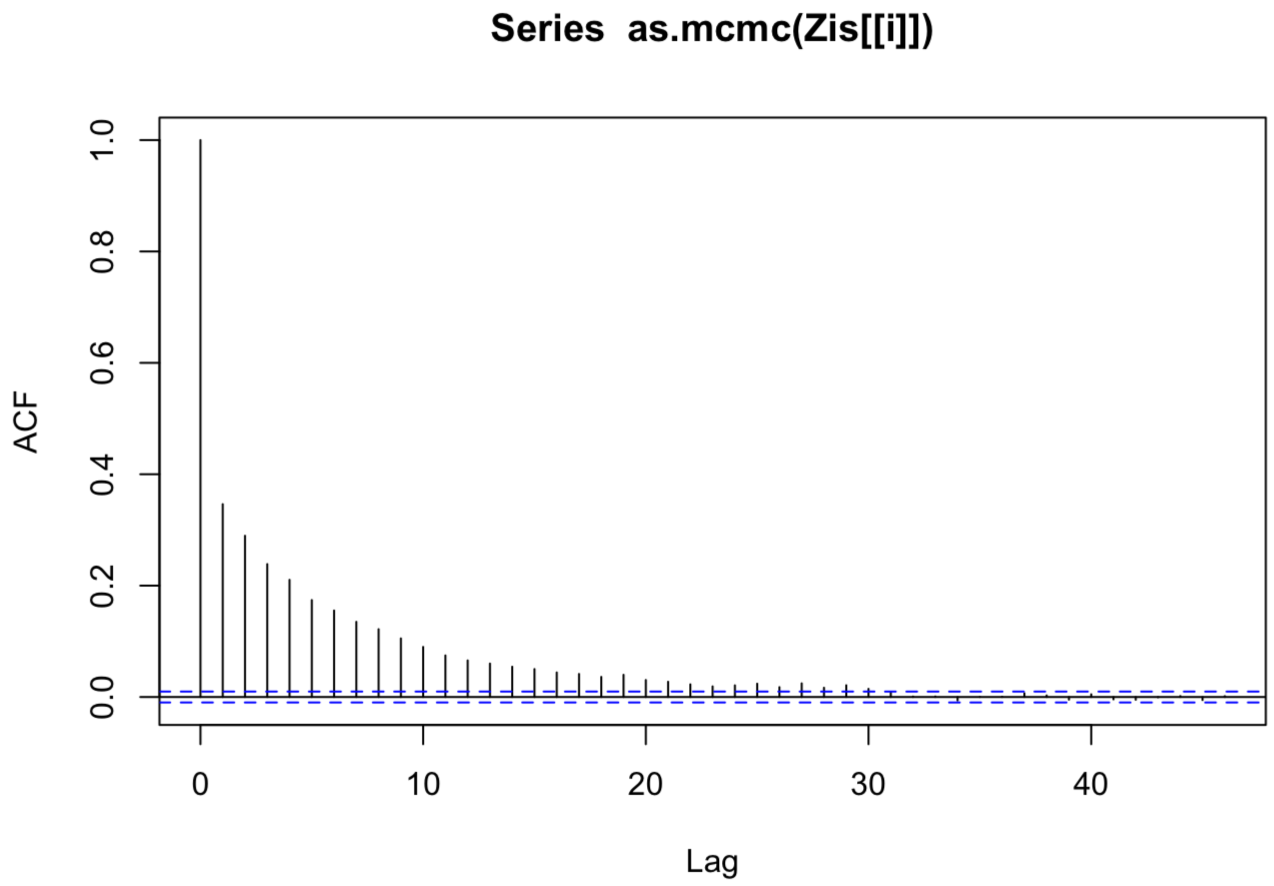

Series as.mcmc(Zis[[i]])
ACF
Lag
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40

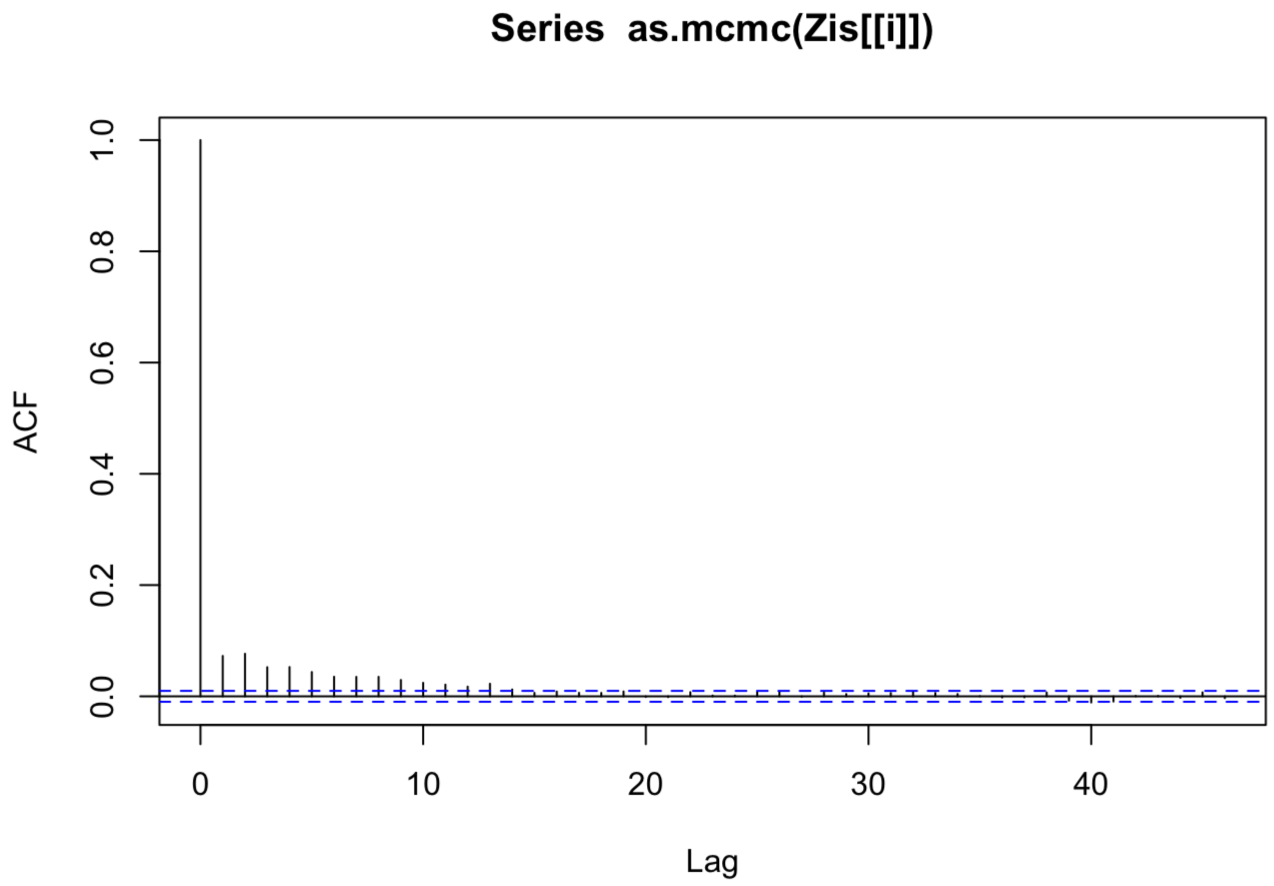

Series as.mcmc(Zis[[i]])
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
Lag

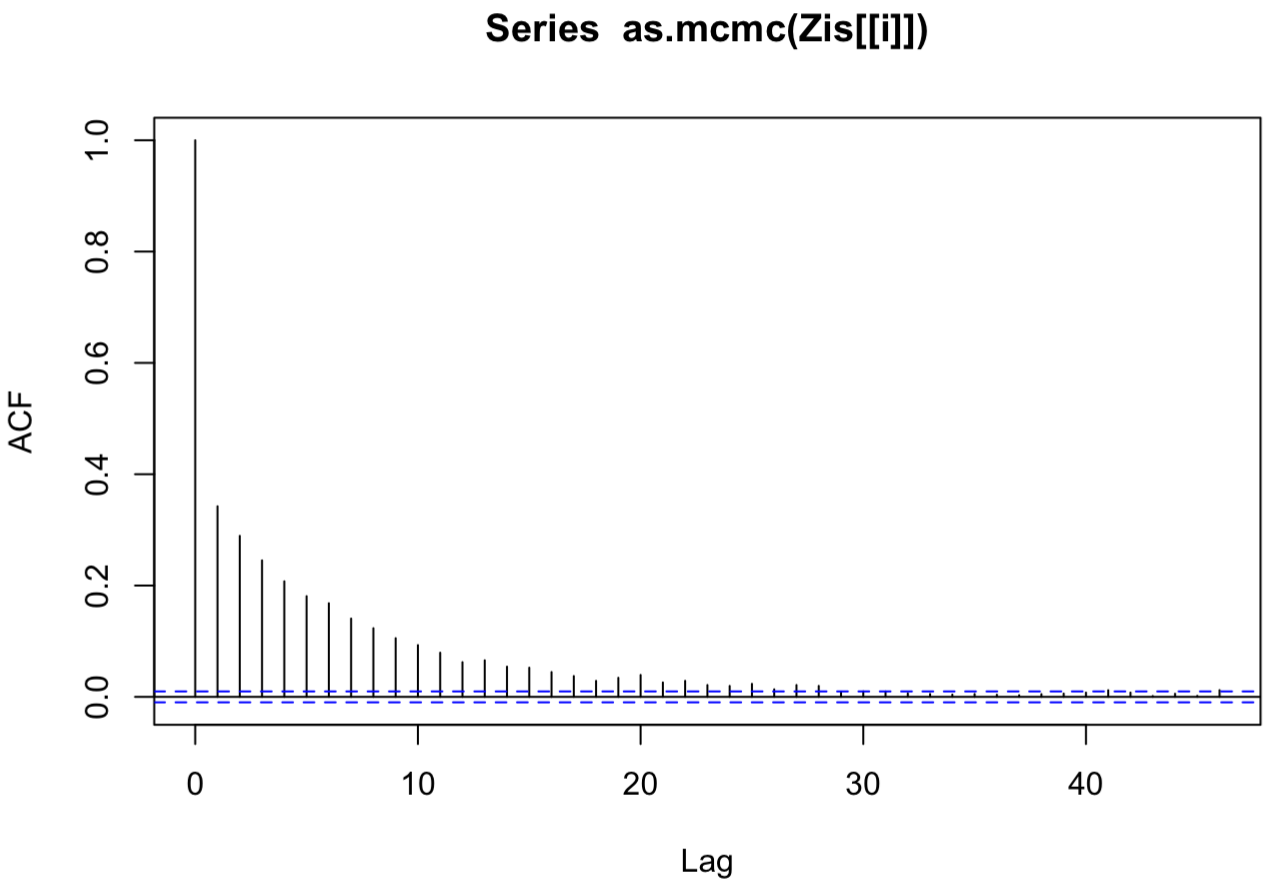

Series as.mcmc(Zis[[i]])
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
Lag

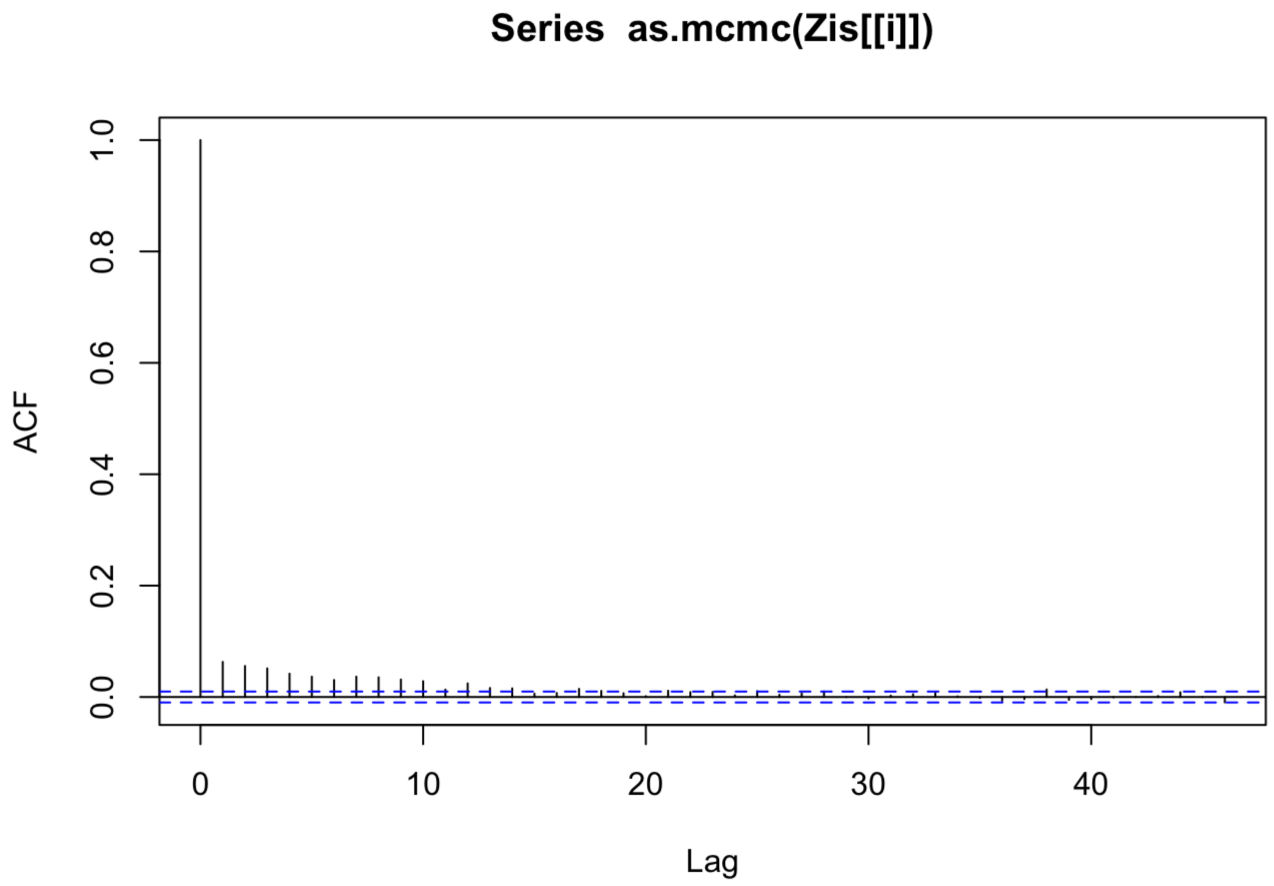

Series as.mcmc(Zis[[i]])
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
Lag

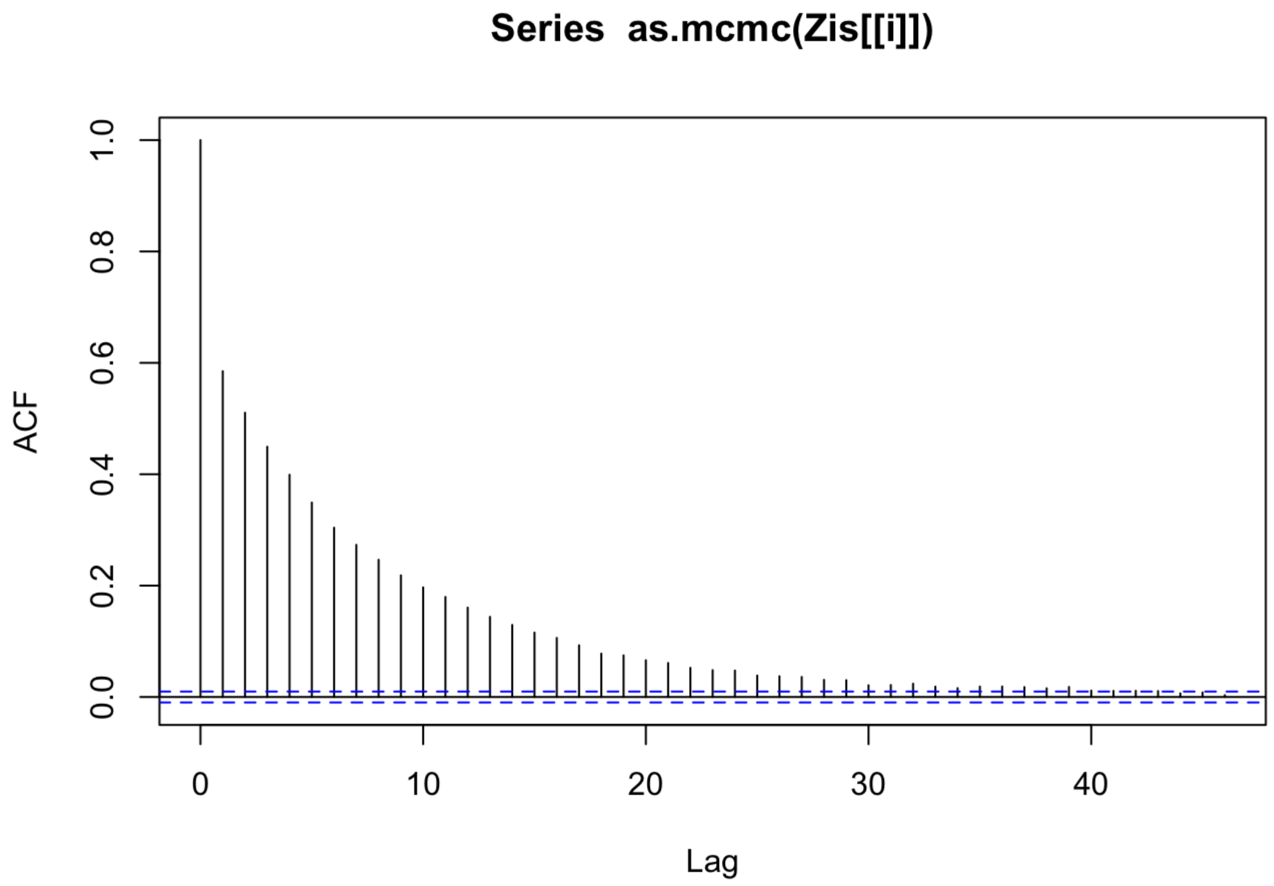

Series as.mcmc(Zis[[i]])
ACF
Lag
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40

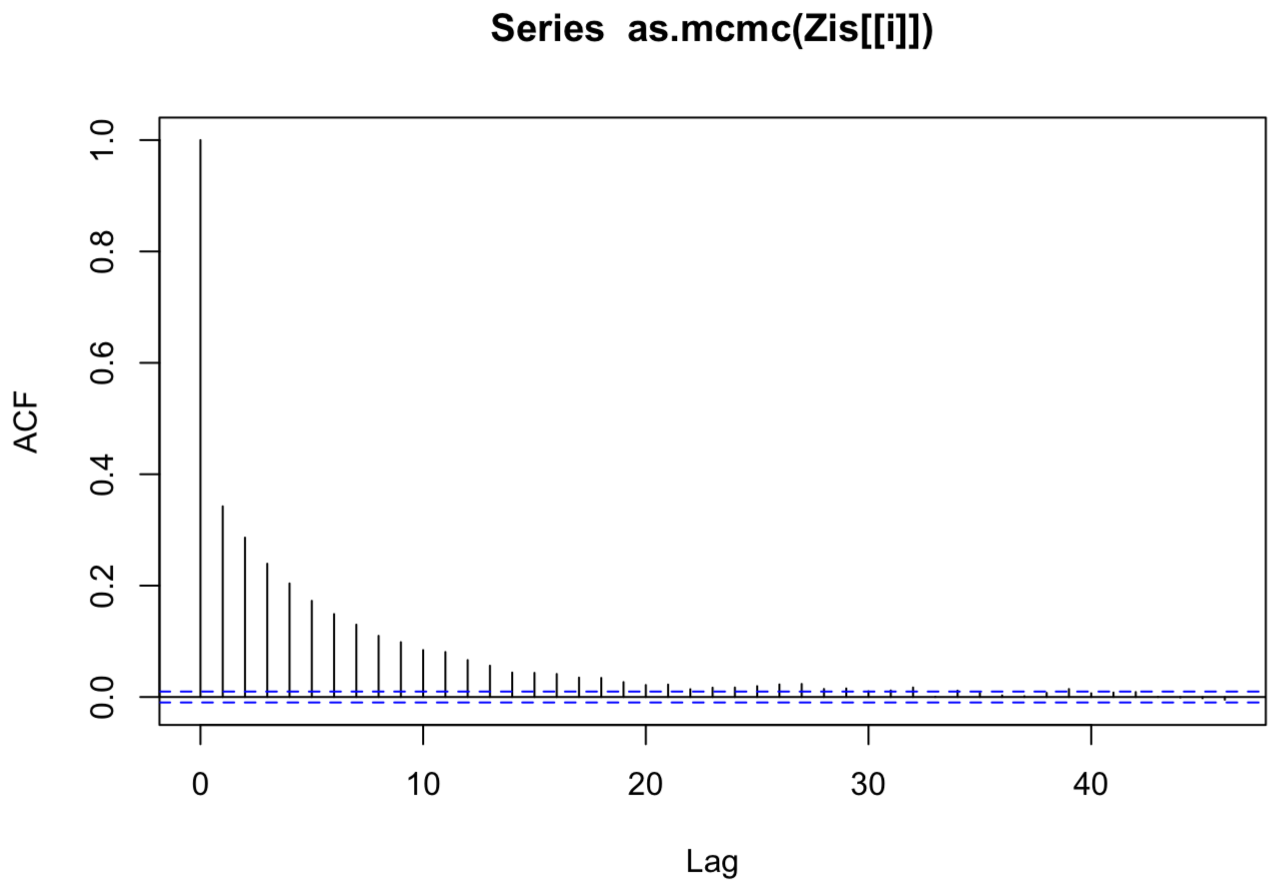

Series as.mcmc(Zis[[i]])
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
Lag

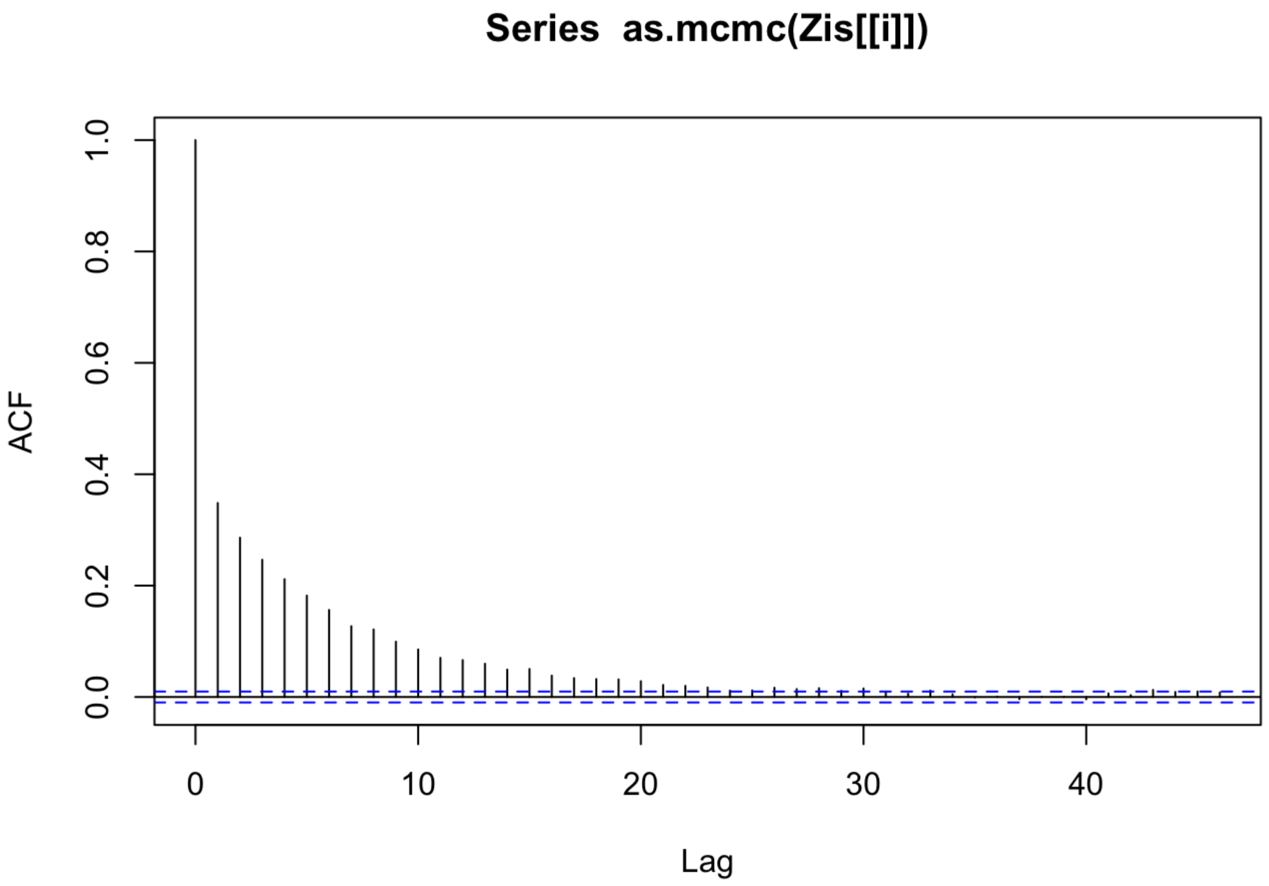

Series as.mcmc(Zis[[i]])
ACF
Lag
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40

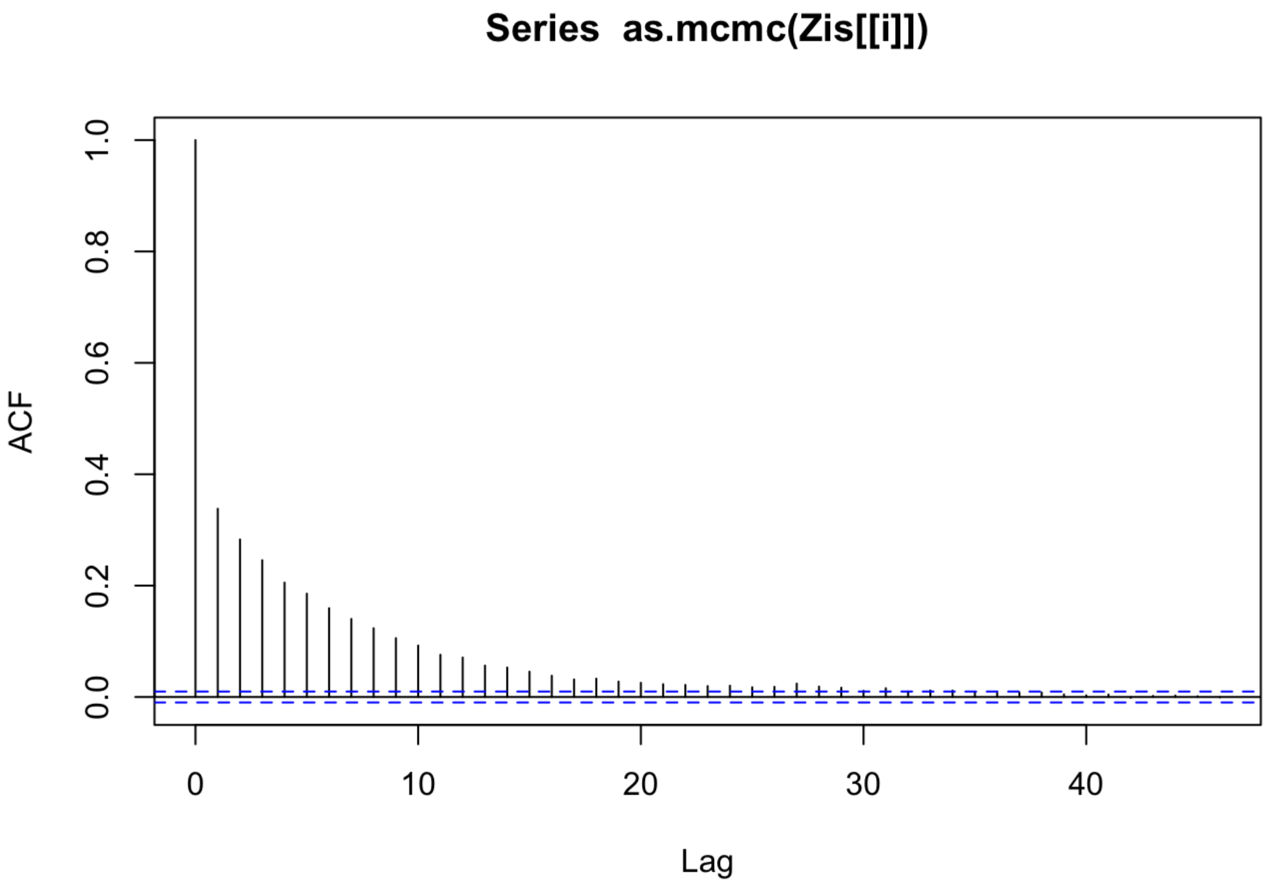

Series as.mcmc(Zis[[i]])
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
Lag

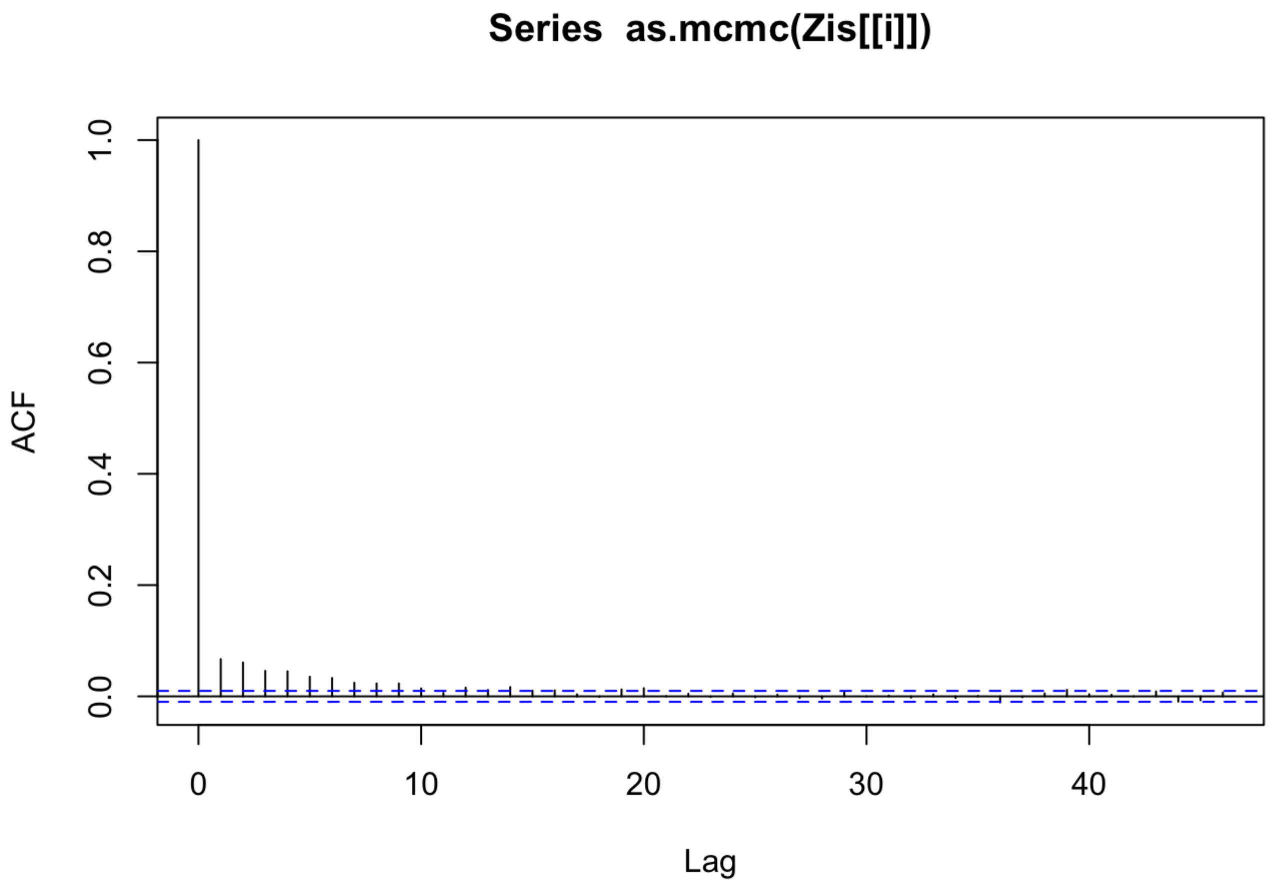

Series as.mcmc(Zis[[i]])
ACF
Lag
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40

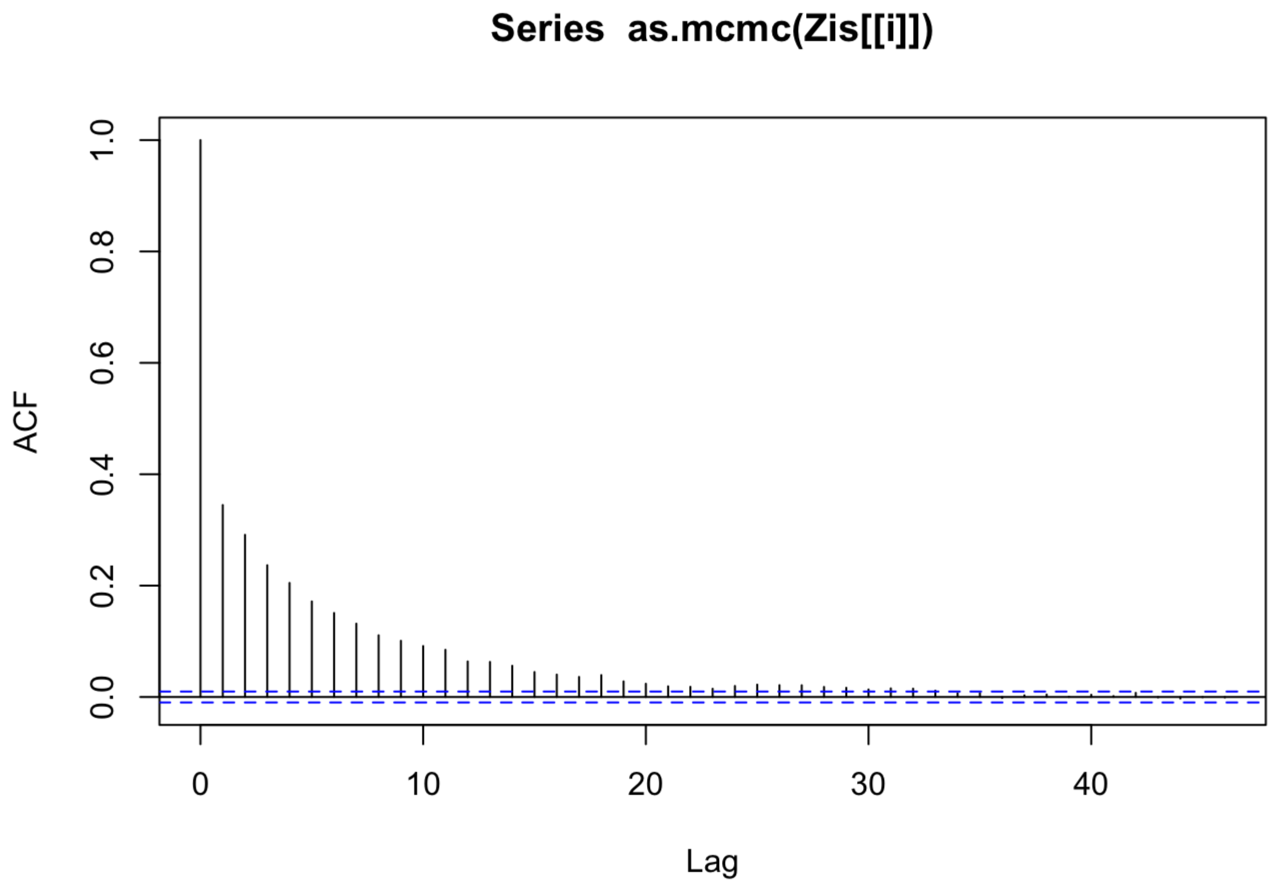

Series as.mcmc(Zis[[i]])
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
Lag

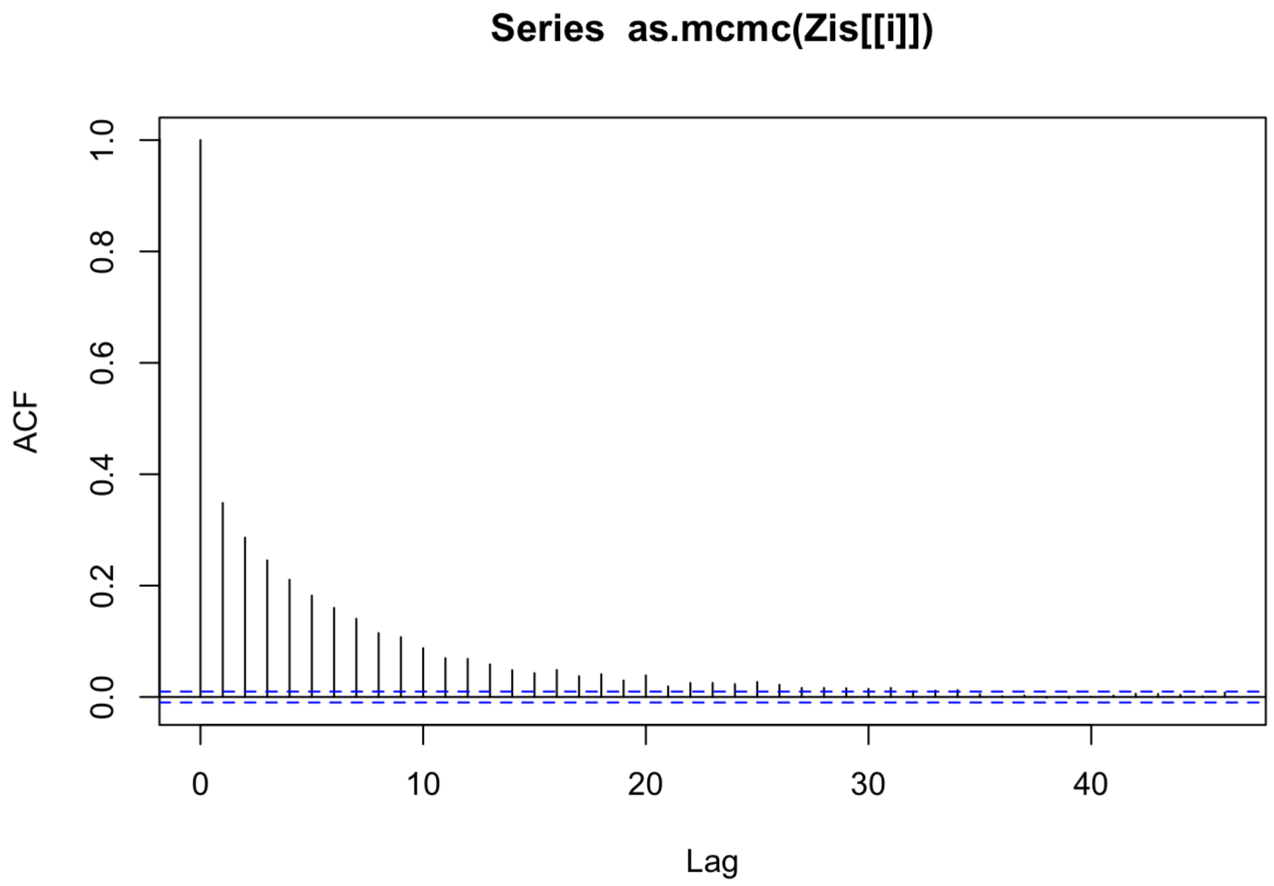

Series as.mcmc(Zis[[i]])
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
Lag

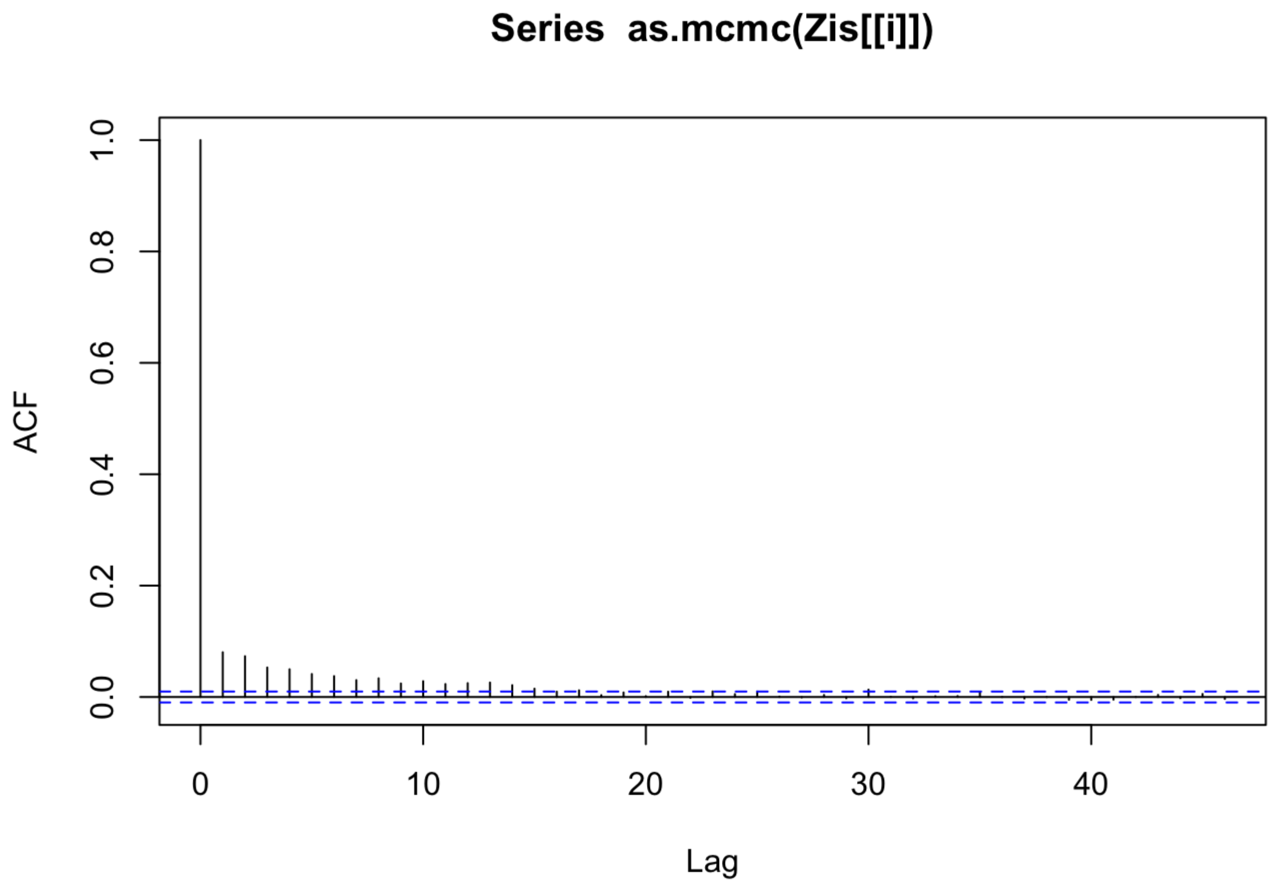

Series as.mcmc(Zis[[i]])
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
Lag

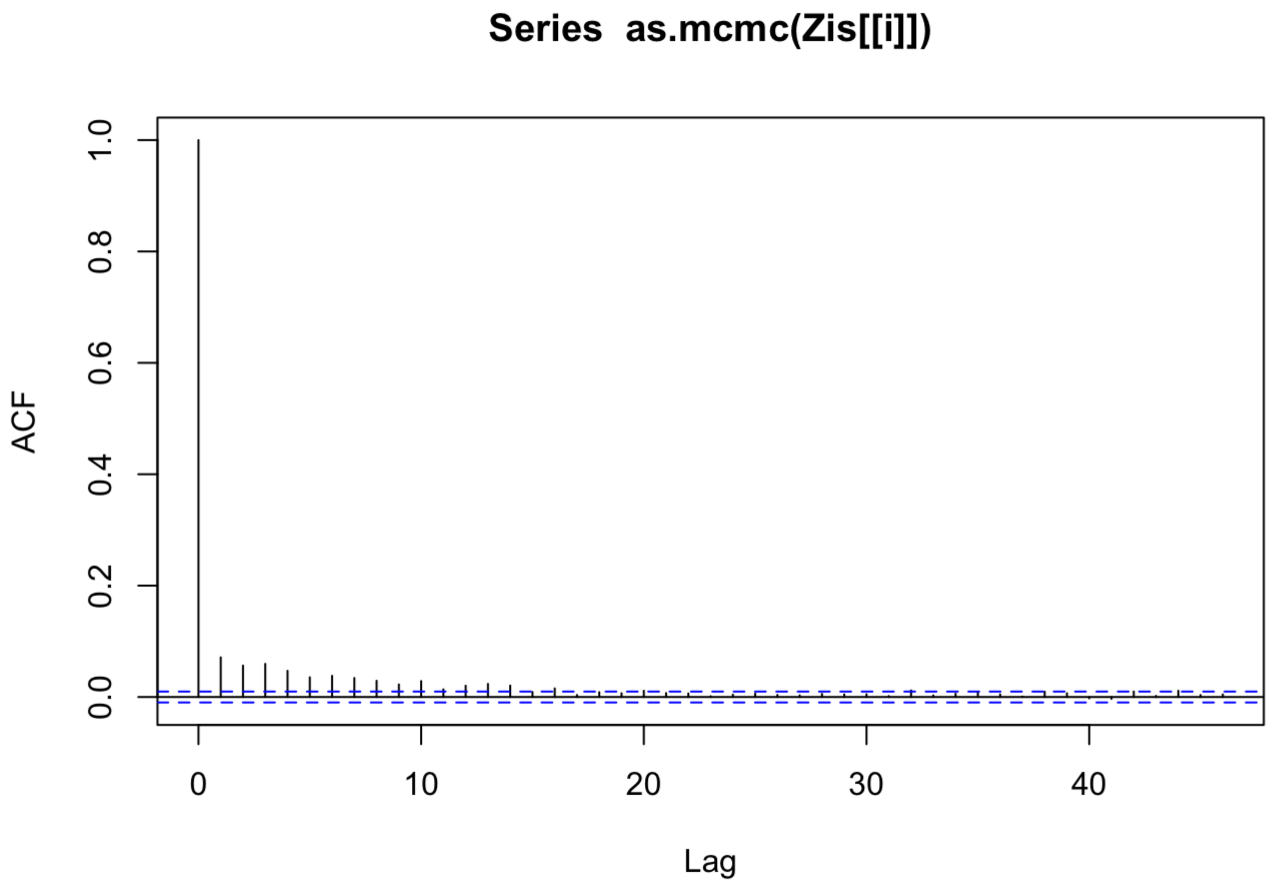

Series as.mcmc(Zis[[i]])
ACF
Lag
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40

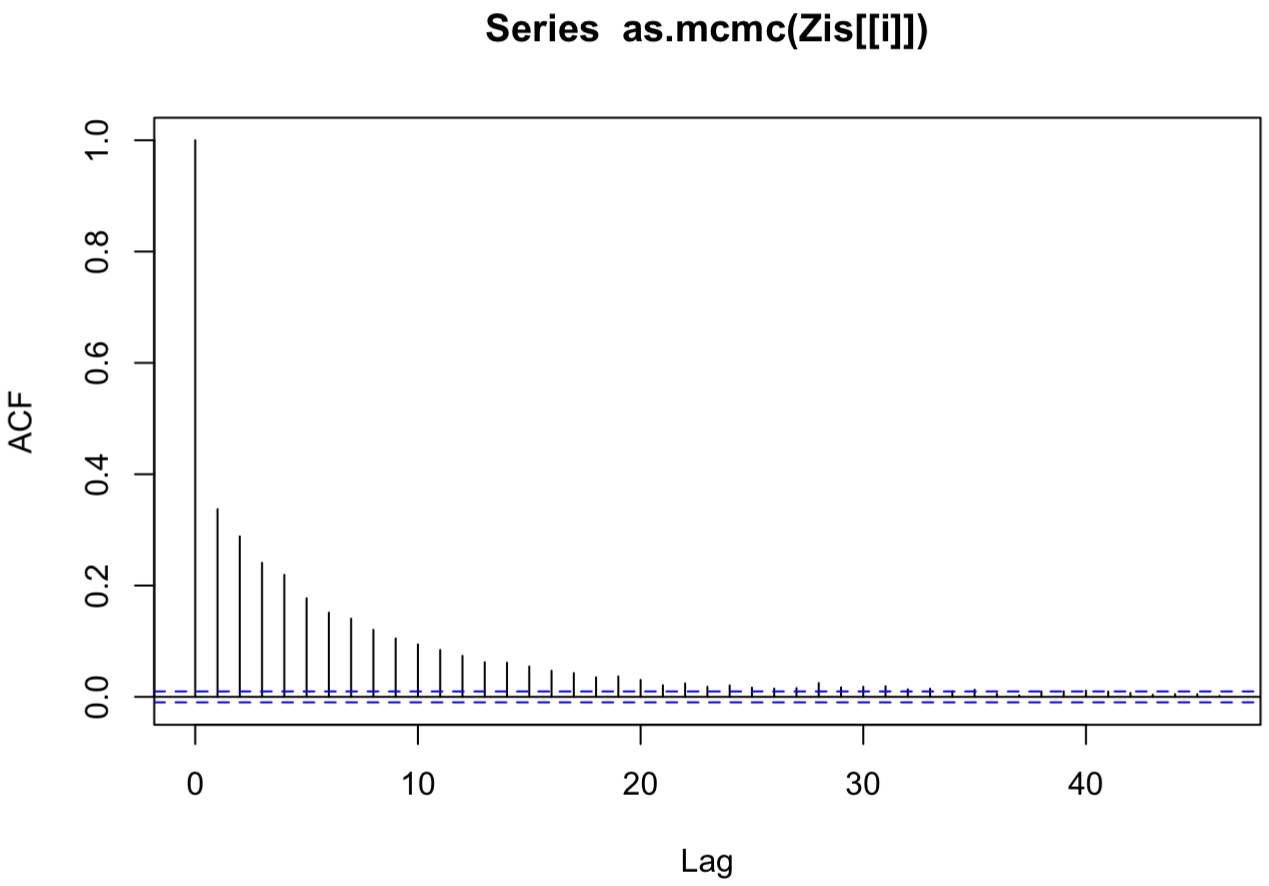

Series as.mcmc(Zis[[i]])
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
Lag

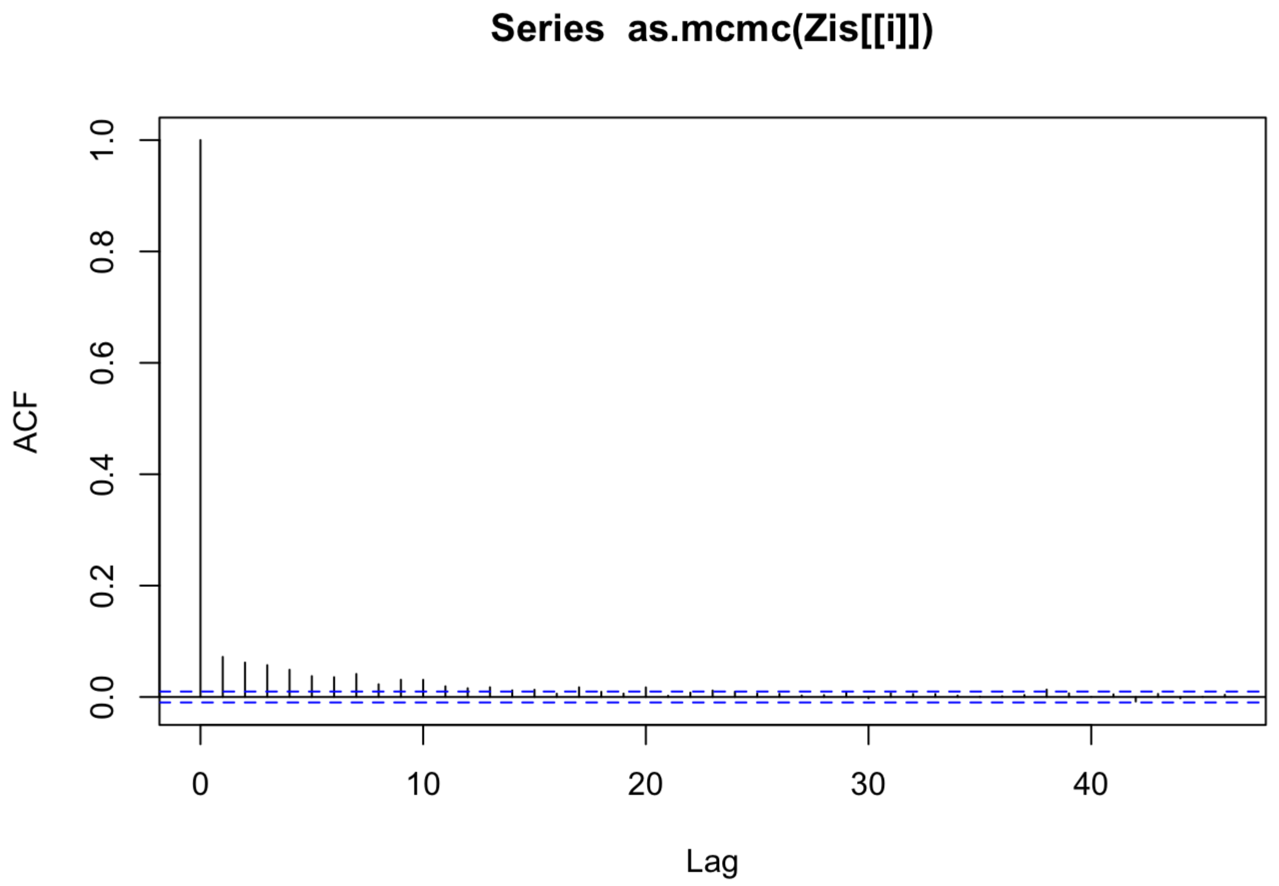

Series as.mcmc(Zis[[i]])
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
Lag

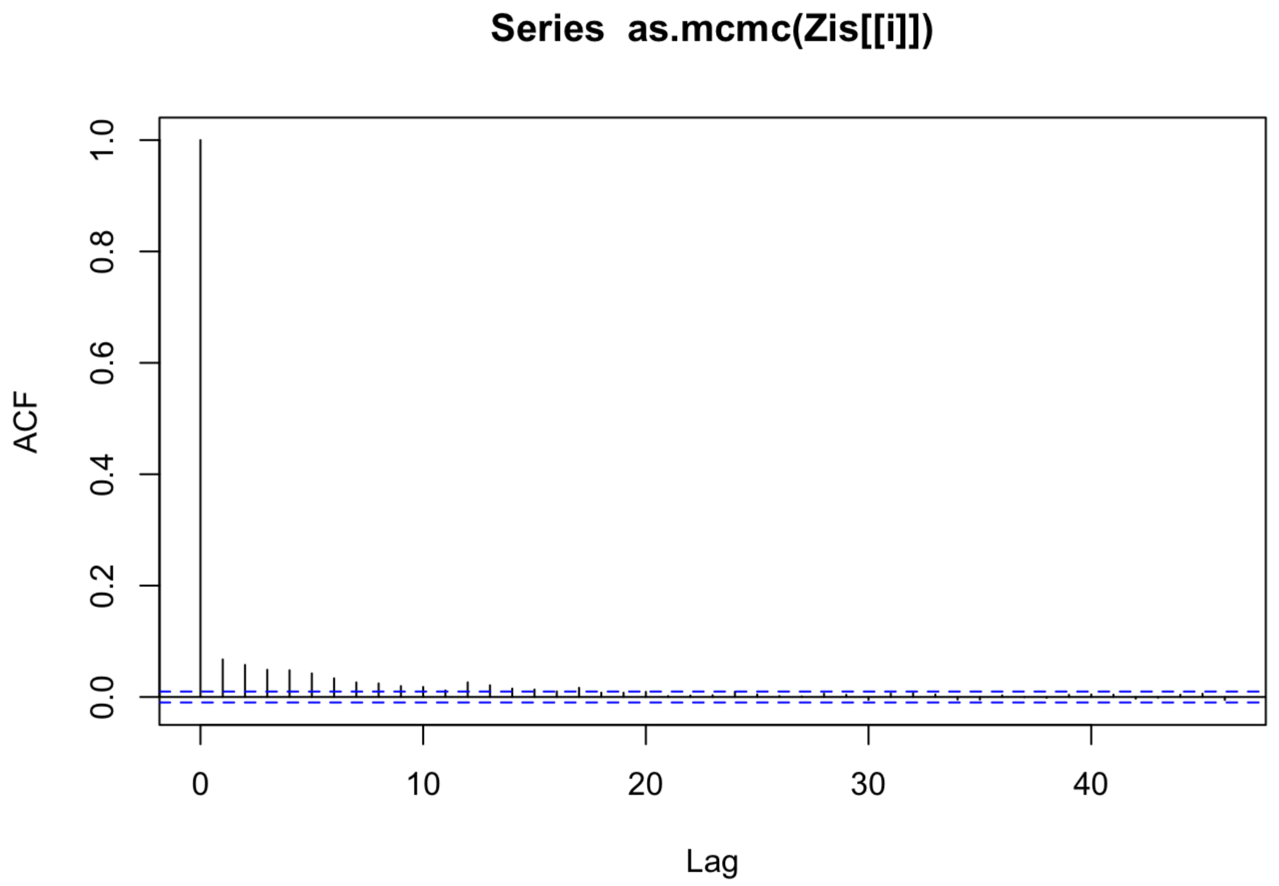

Series as.mcmc(Zis[[i]])
ACF
Lag
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40

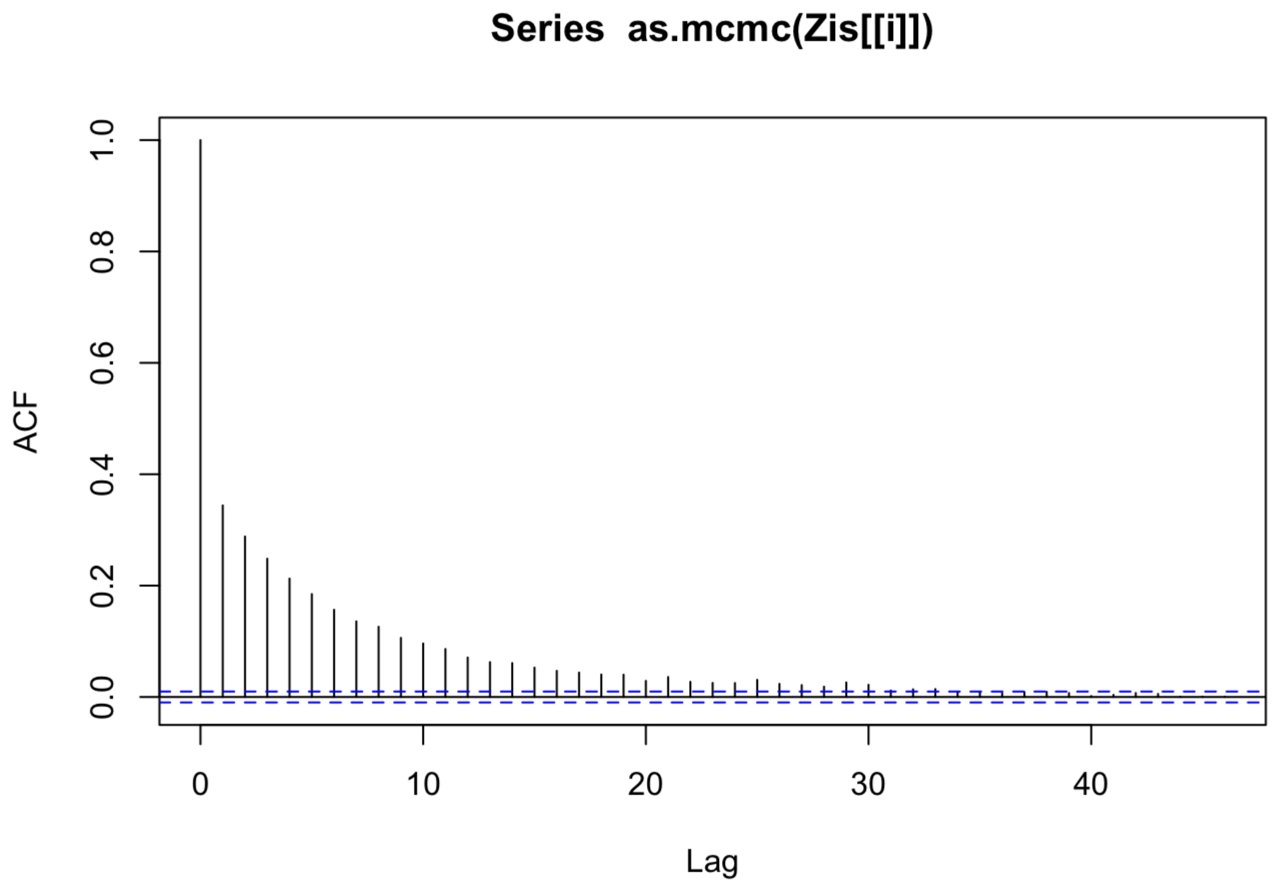

Series as.mcmc(Zis[[i]])
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
Lag

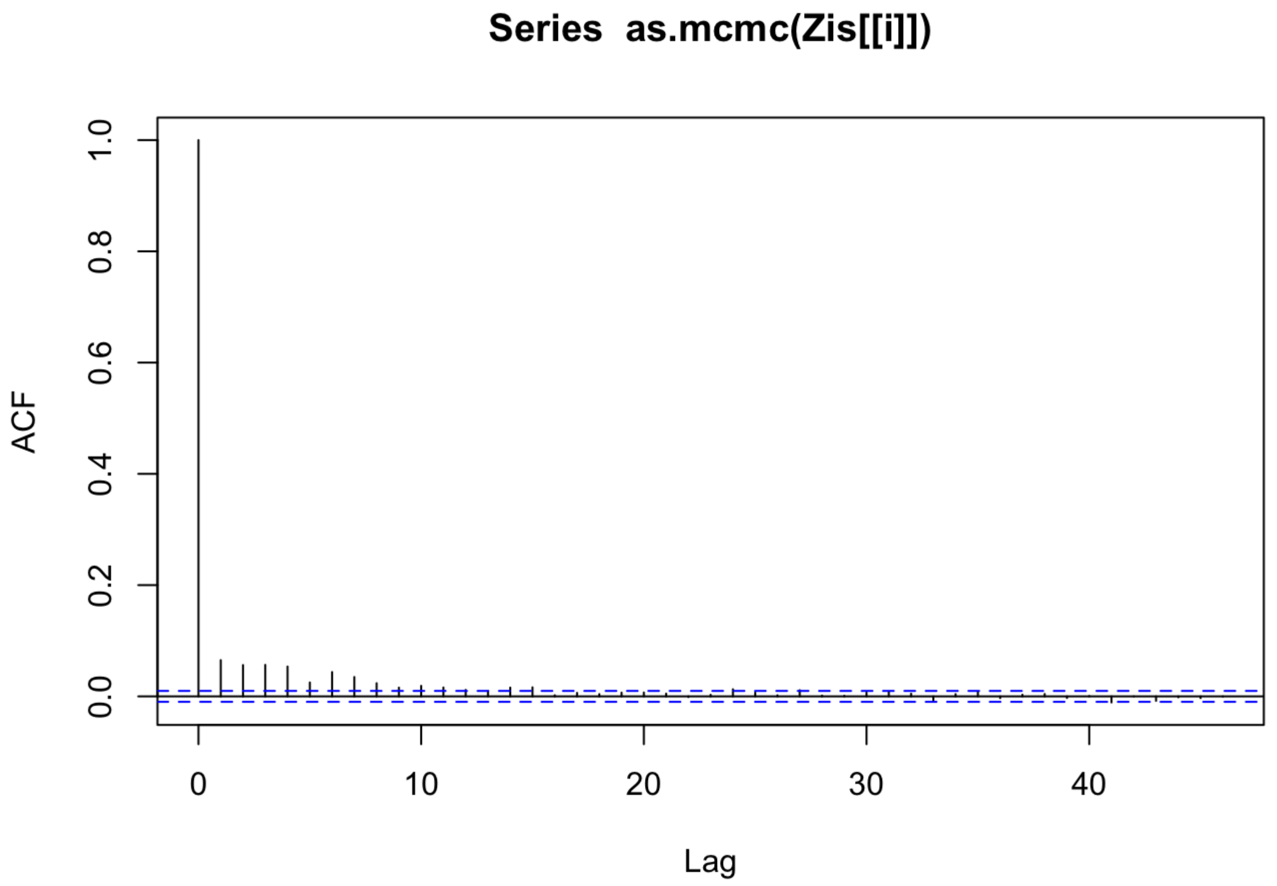

Series as.mcmc(Zis[[i]])
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
Lag


Series as.mcmc(Zis[[i]])
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
Lag


Series as.mcmc(Zis[[i]])
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
Lag


Series as.mcmc(Zis[[i]])
ACF
Lag
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40


Series as.mcmc(Zis[[i]])
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
Lag



According to the autocorrelation plots, all values decreases and converges to 0 as we perform sufficient number of simulations , indicating that the mixing of the Markov Chain has converged to a steady distribution.

Part d

```
# 95% CI for beta
CI <- quantile(BETA, c(0.025, 0.975))
print(CI)
```

```
##       2.5%      97.5%
## -2.1479751  0.7194671
```

```
# Pr(beta > 0 | y, x)
prob <- mean(BETA > 0)
print(prob)
```

```
## [1] 0.1723
```