STAT 466 HW 8

Brody Anderson

2024-03-08

Question 1

Reanalyze the supercomputer data in Example 4.2 using a uniform prior distribution on the interval (0, 20) for the mean monthly number of failures λ. Compare the results with those obtained in Example 4.2 in terms of the point estimate, credible interval, and length of the credible interval.

```
superdata <- c(1, 5, 1, 4, 2, 3, 1, 3, 6, 4, 4, 4, 2, 3, 2, 2, 4, 5, 5, 2, 5, 3, 2, 2, 3, 1,
               1, 2, 5, 1, 4, 1, 1, 1, 2, 1, 3, 2, 5, 3, 5, 2, 5, 1, 1, 5, 2)

#  GoF[i] <- pgamma(y[i], alpha, beta);

Super_model <- "model {
  for (i in 1:length(superdata)) {
  superdata[i] ~ dgamma(alpha, beta);
  }

  alpha ~ dunif(0,20)
  beta ~ dunif(0,20)

}"
```

```
Super.sim <- jags( data = c('superdata'), parameters.to.save = c('alpha', 'beta'),
  model.file = textConnection(Super_model),
  n.iter = 22000,
  n.burnin=2000,
  n.chains=5,
  n.thin=1
)
```

```
## module glm loaded
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 47
##    Unobserved stochastic nodes: 2
##    Total graph size: 51
##
## Initializing model
```

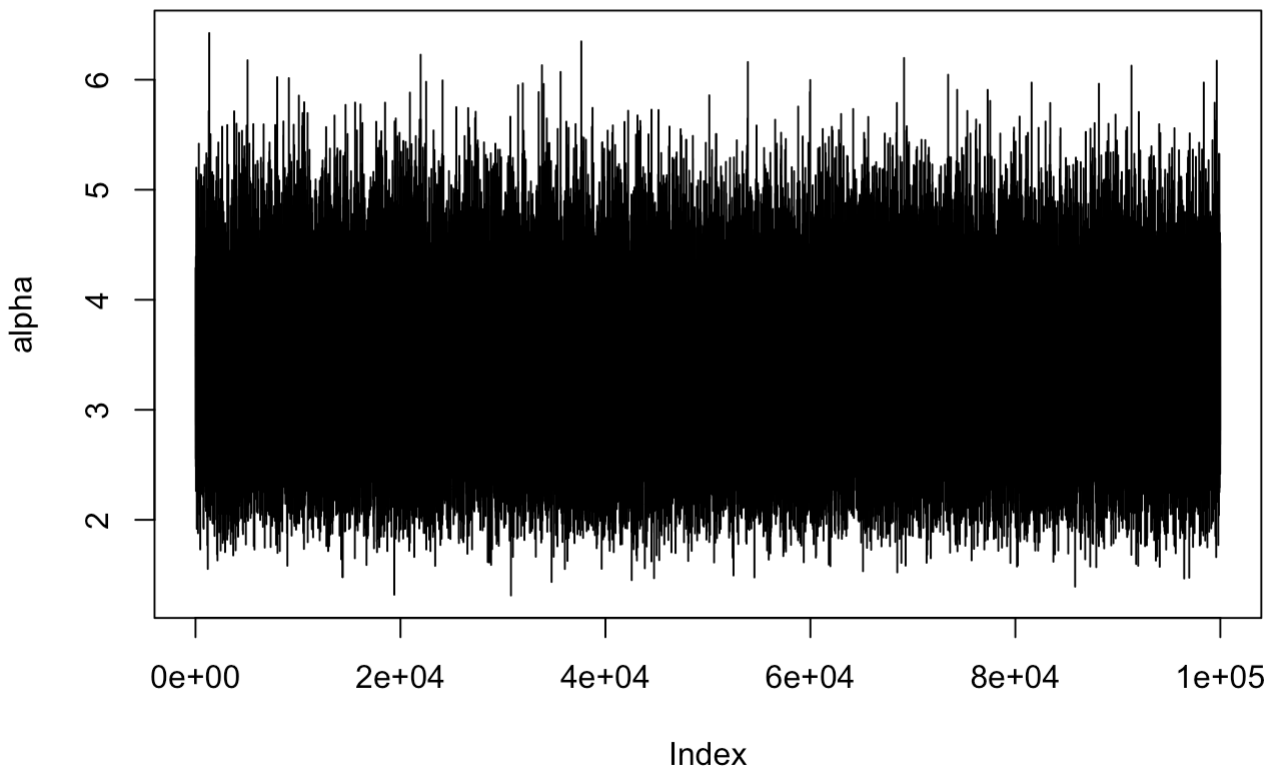
```
#head(Super.sim$BUGSoutput$sims.matrix) #100,000 samples
length(Super.sim$BUGSoutput$sims.matrix[,1])
```

```
## [1] 100000
```

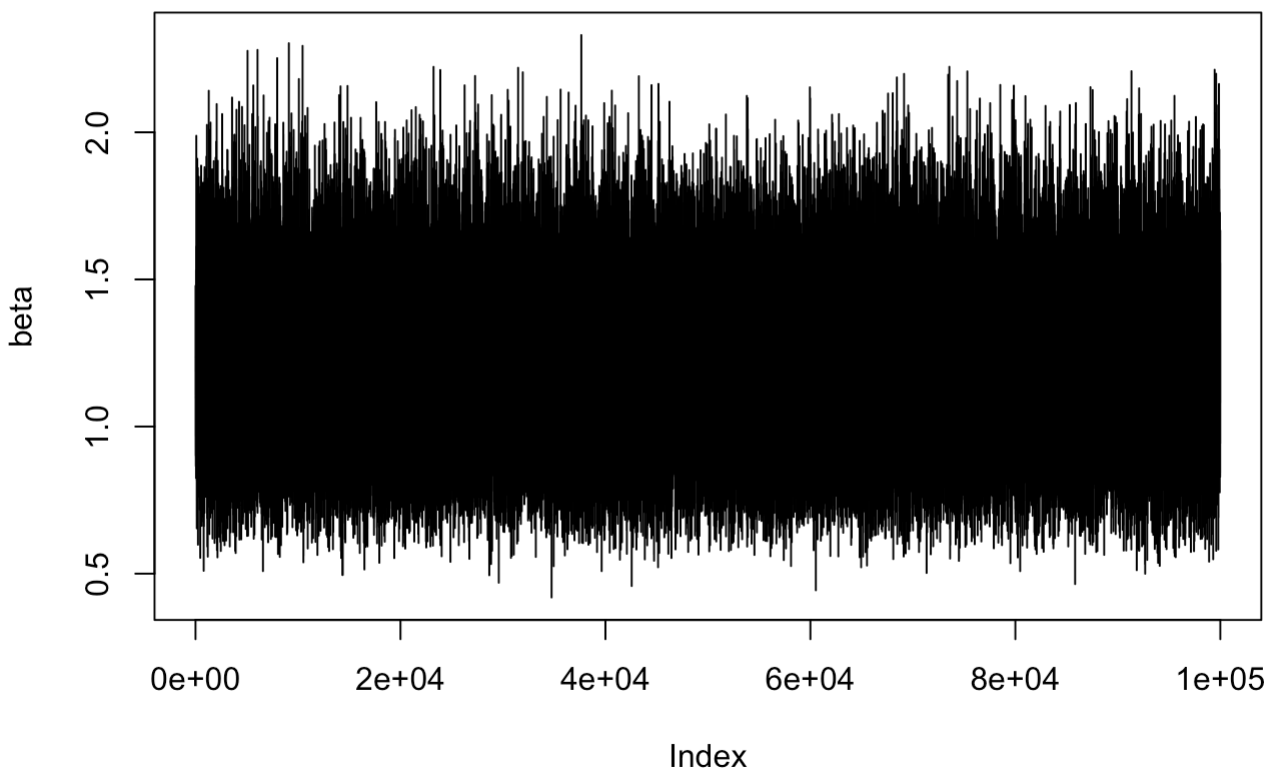
```
alpha <- Super.sim$BUGSoutput$sims.matrix[,1]
beta <- Super.sim$BUGSoutput$sims.matrix[,2]

#jags.matrix <- Super.sim$BUGSoutput$sims.matrix[, 48:50]
#head(jags.matrix)

q1mean <- mean(alpha / beta)
samp <- alpha / beta
q1credint <- quantile(samp, c(0.025, 0.975))

gelman.diag(Super.sim$BUGSoutput,multivariate=F)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## alpha             1          1
## beta              1          1
## deviance          1          1
```

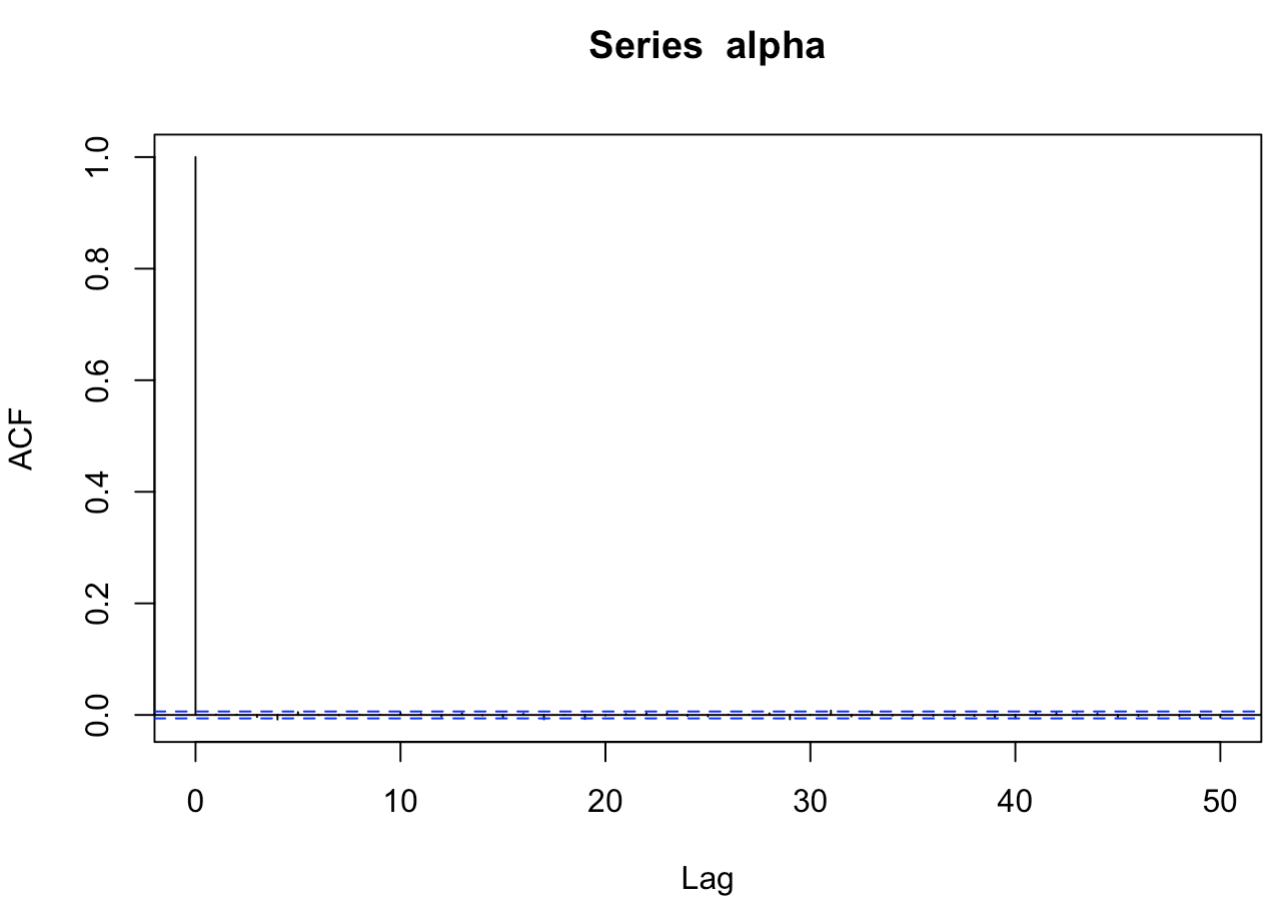
```
plot(alpha,type='l')
```



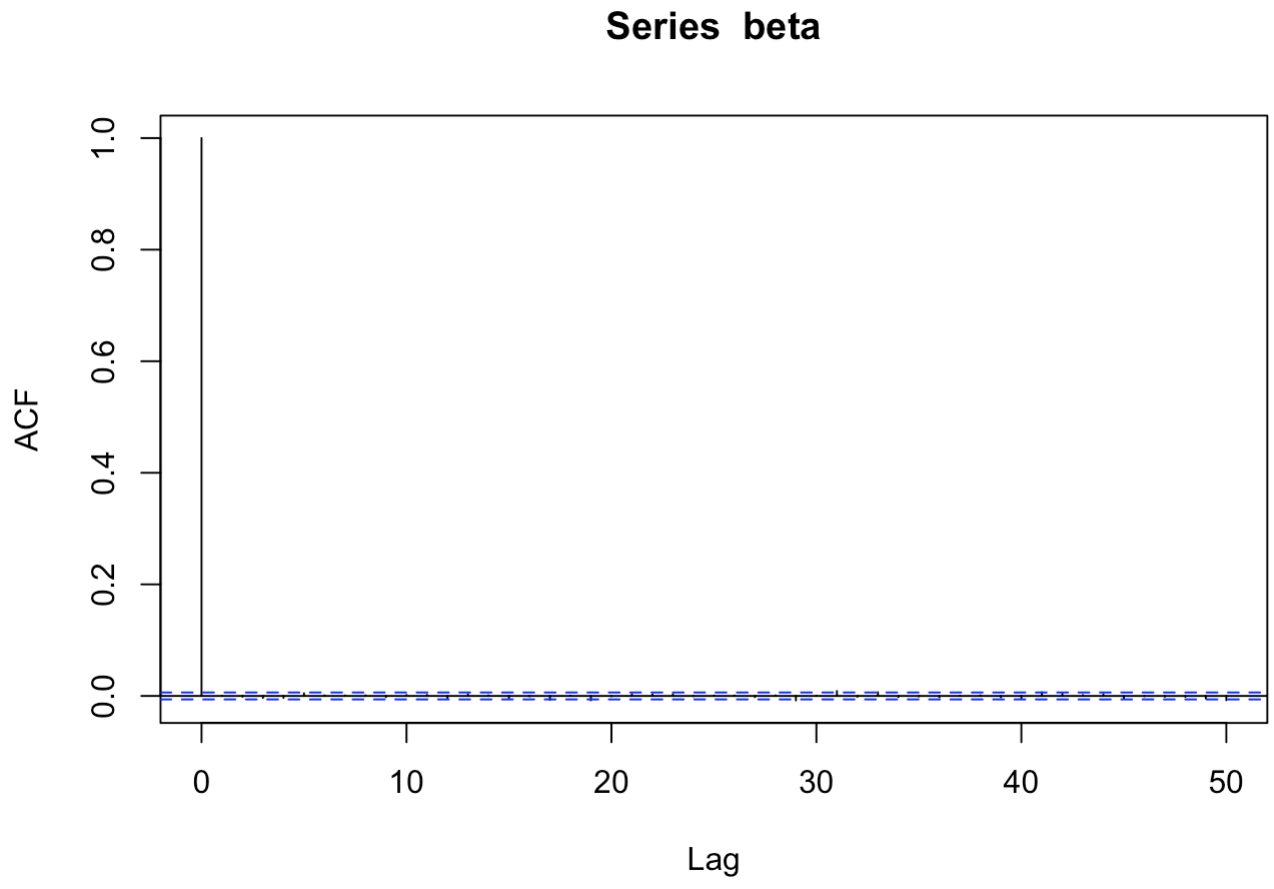

```
plot(beta, type='l')
```




```
acf(alpha)
```



```
acf(beta)
```



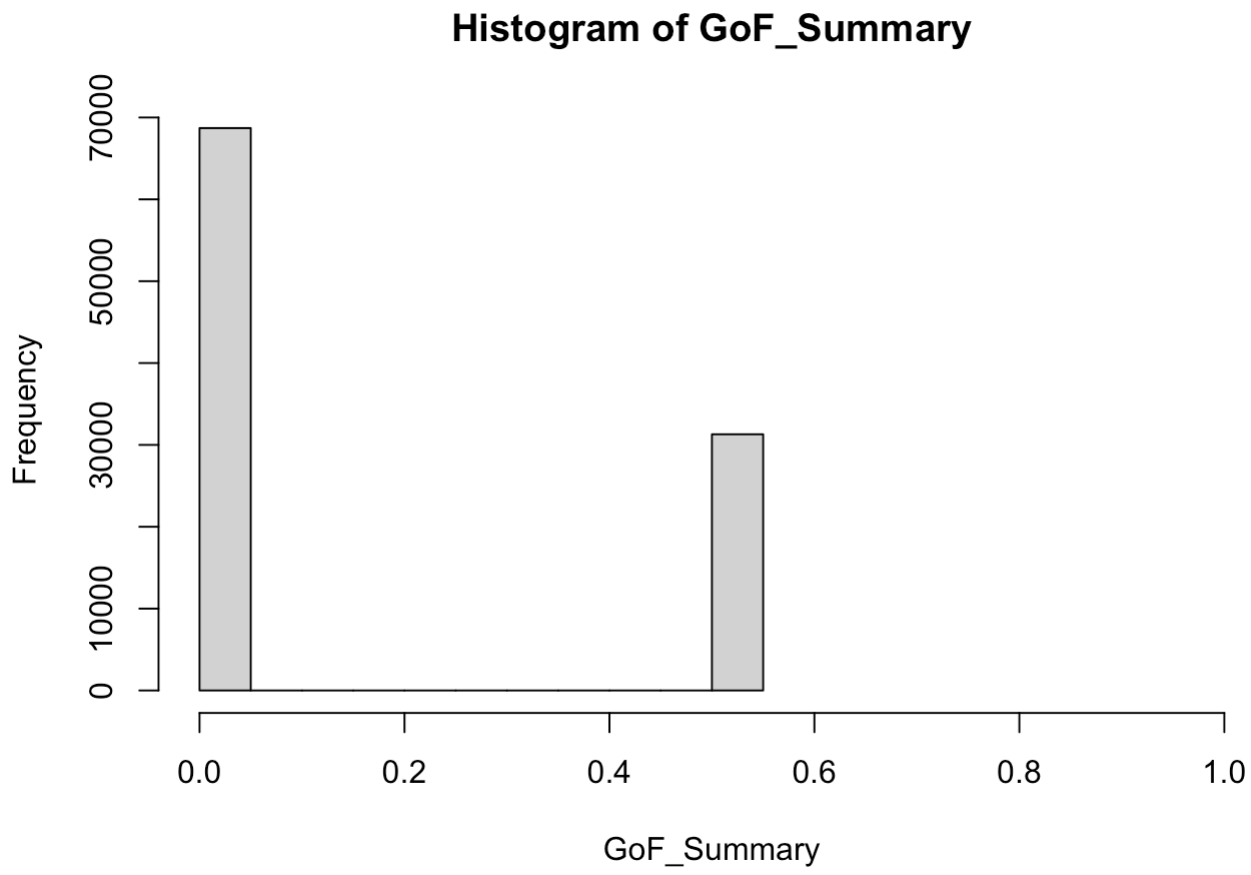

```
Super.sim$BUGSoutput$DIC
```

```
## [1] 170.2403
```

```
ggchains <- Super.sim$BUGSoutput$sims.matrix
effectiveSize(Super.sim$BUGSoutput$sims.matrix)
```

```
##    alpha     beta deviance
## 101131.9 100000.0 100000.0
```

```
# Goodness of Fit
GoF <- matrix(NA,ncol=length(superdata),nrow=length(ggchains[,3]))
for (i in 1:length(ggchains[,3])) {
  GoF[i,] <- pgamma(superdata, alpha[i], beta[i])
}

# Function requires fitted quantiles and returns a p-value
GoF_Test <- function(fitted_quantiles) {
  n <- length(fitted_quantiles)
  K <- round((n)^(0.4))
  mK <- table(cut(fitted_quantiles,(0:K)/K))
  np <- n/K
  RB <- sum(((mK-np)^2)/np)
  return(1-pchisq(RB,K-1))
}

# Calculating the p-values for each posterior model
GoF_Summary <- apply(GoF,1,GoF_Test)

# Histogram of posterior model p-values
hist(GoF_Summary,xlim=c(0,1))
```




```
# Percent of posterior models with p-value less than 0.05
mean(GoF_Summary < 0.05)
```

```
## [1] 0.68705
```

Question 2

Like Example 4.2, analyze the failure count data for SMP 21 in Table 6.3 assuming a Poisson distribution. Separately analyze the failure count data from SMP 1 and compare the mean number of failures λ for these two SMPs.

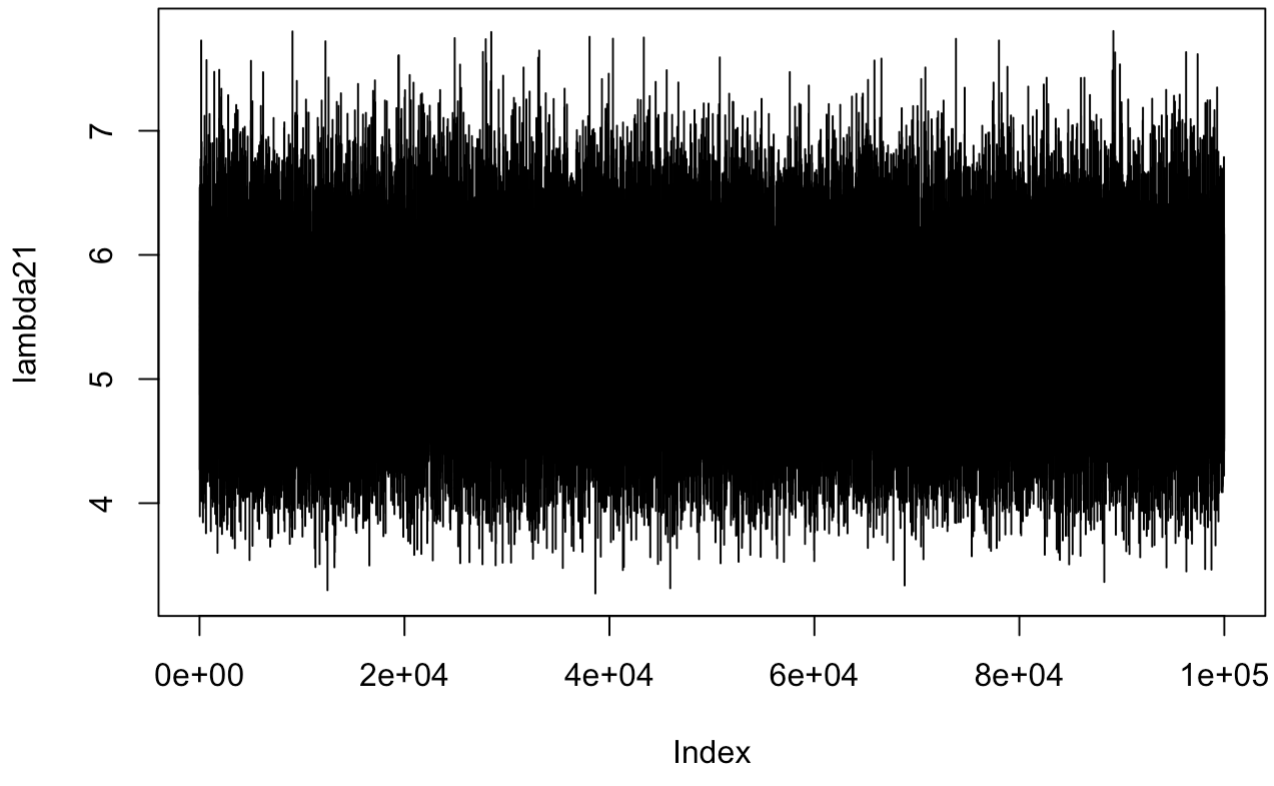
```
smp21 <- c(10, 5, 3, 6, 6, 4, 10, 5, 3, 8, 3, 2, 8, 2, 5)
smp1 <- c(1, 5, 2, 1, 0, 1, 1, 2, 1, 0, 1, 0, 1, 2, 5)

# SMP 21
Smp21_model <- "model {
  for (i in 1:15) {
  smp21[i] ~ dpois(lambda);
  }

  lambda ~ dgamma(5, 1)

}"

Smp21.sim <- jags( data = c('smp21'), parameters.to.save = c('lambda'),
  model.file = textConnection(Smp21_model),
  n.iter = 22000,
  n.burnin=2000,
  n.chains=5,
  n.thin=1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 15
##    Unobserved stochastic nodes: 1
##    Total graph size: 18
##
## Initializing model
```

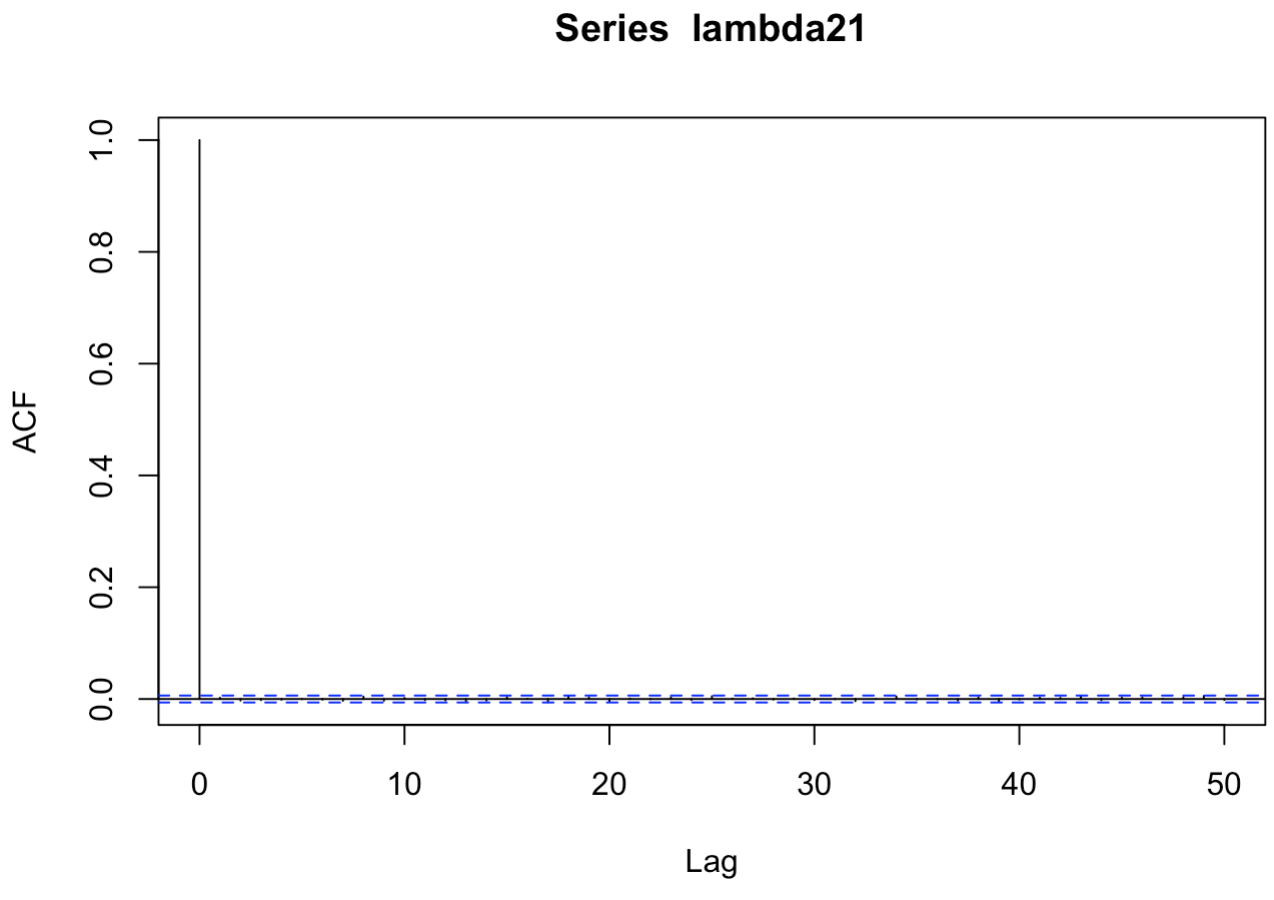
```
#head(Smp21.sim$BUGSoutput$sims.matrix) #100,000 samples

lambda21 <- Smp21.sim$BUGSoutput$sims.matrix[,2]

gelman.diag(Smp21.sim$BUGSoutput,multivariate=F)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## deviance          1          1
## lambda            1          1
```

```
plot(lambda21,type='l')
```




```
acf(lambda21)
```



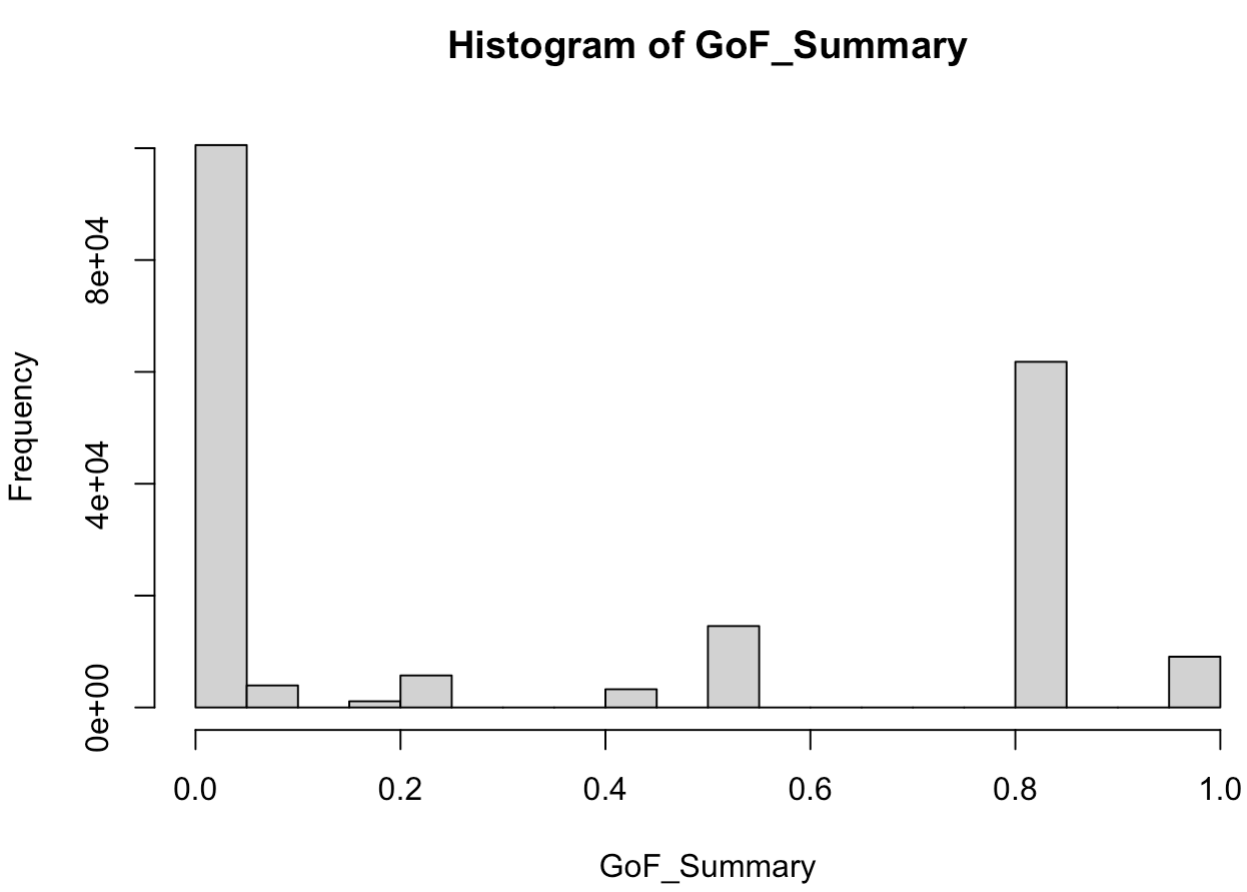

```
Smp21.sim$BUGSoutput$DIC
```

```
## [1] 71.75801
```

```
effectiveSize(ggchains)
```

```
##    alpha     beta deviance
## 101131.9 100000.0 100000.0
```

```
mcmcChains <- Smp21.sim$BUGSoutput$sims.matrix

# Goodness of Fit
GoF <- matrix(NA,ncol=length(smp21),nrow=length(mcmcChains))
for (i in 1:length(mcmcChains)) {
  GoF[i,] <- runif(length(smp21),ppois(smp21-1, mcmcChains[i]),ppois(smp21, mcmcChains[i]))
}

GoF_Test <- function(fitted_quantiles) {
  n <- length(fitted_quantiles)
  K <- round((n)^(0.4))
  mK <- table(cut(fitted_quantiles,(0:K)/K))
  np <- n/K
  RB <- sum(((mK-np)^2)/np)
  return(1-pchisq(RB,K-1))
}

GoF_Summary <- apply(GoF,1,GoF_Test)

hist(GoF_Summary,xlim=c(0,1))
```




```
mean(GoF_Summary < 0.05)
```

```
## [1] 0.502755
```

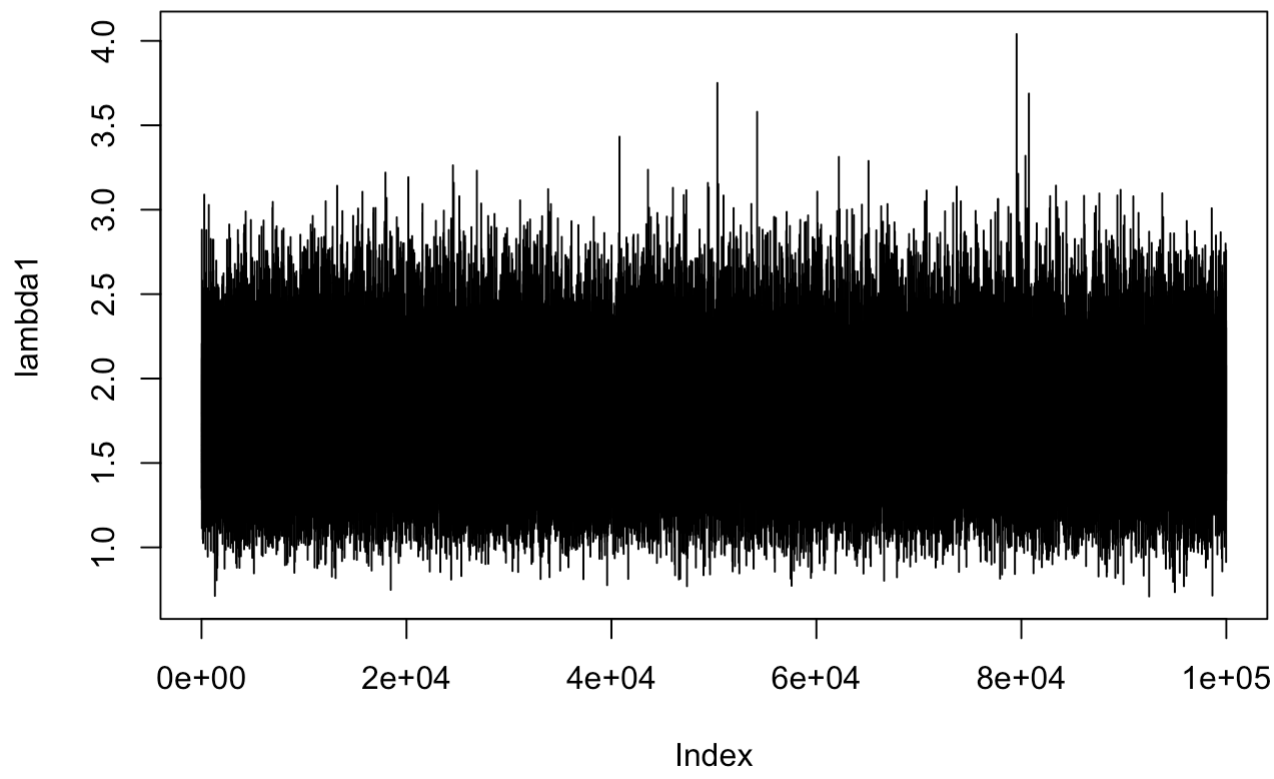
```
# SMP 1
Smp1_model <- "model {
  for (i in 1:15) {smp1[i] ~ dpois(lambda);
  }

  lambda ~ dgamma(5, 1)

}"

Smp1.sim <- jags( data = c('smp1'), parameters.to.save = c('lambda'),
  model.file = textConnection(Smp1_model),
  n.iter = 22000,
  n.burnin=2000,
  n.chains=5,
  n.thin=1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 15
##    Unobserved stochastic nodes: 1
##    Total graph size: 18
##
## Initializing model
```

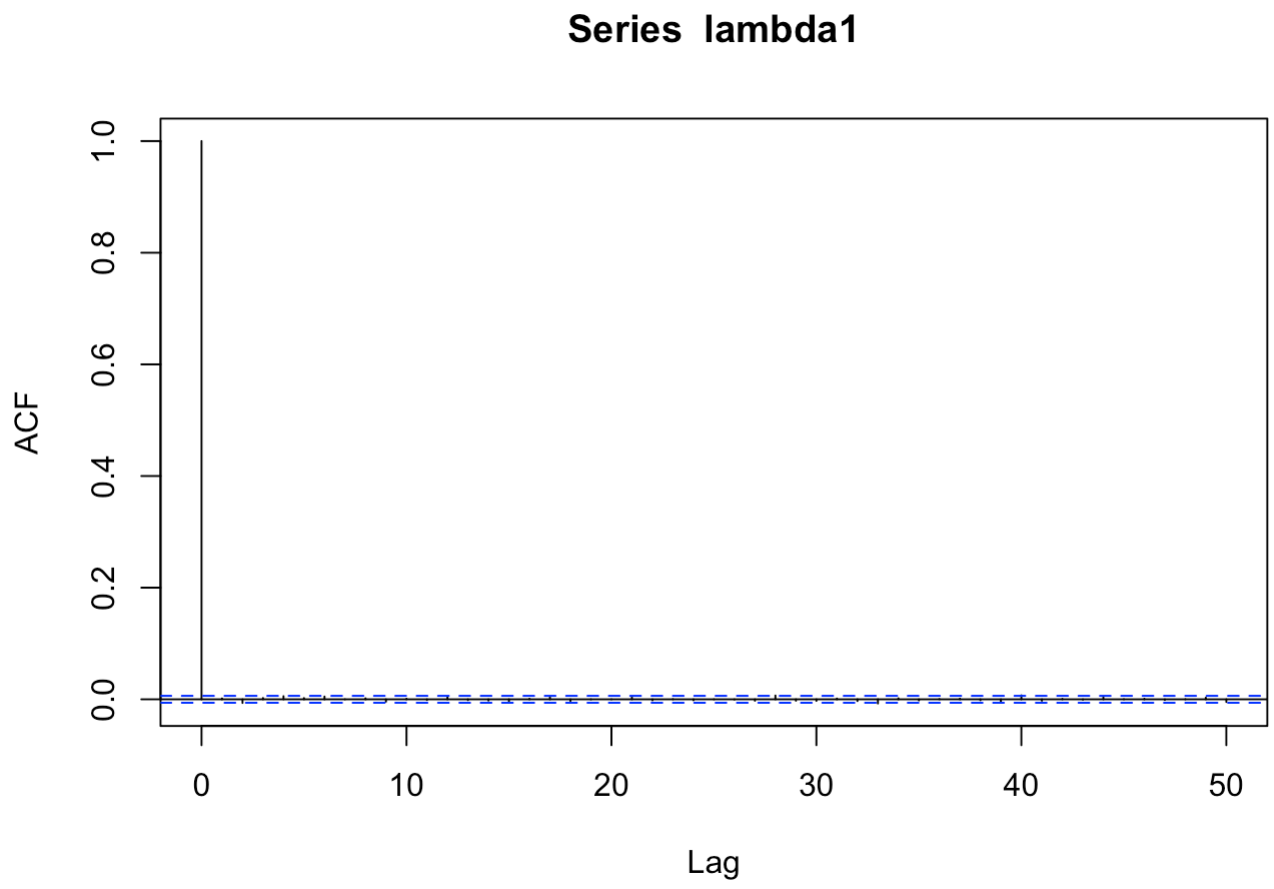
```
#head(Smp1.sim$BUGSoutput$sims.matrix) #100,000 samples
lambda1 <- Smp1.sim$BUGSoutput$sims.matrix[,2]

gelman.diag(Smp1.sim$BUGSoutput,multivariate=F)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## deviance          1          1
## lambda            1          1
```

```
plot(lambda1,type='l')
```




```
acf(lambda1)
```




```
Smp1.sim$BUGSoutput$DIC
```

```
## [1] 52.33798
```

```
ggchains <- Smp1.sim$BUGSoutput$sims.matrix
effectiveSize(ggchains)
```

```
## deviance   lambda
## 100000.0 100980.3
```

```
mean(Smp21.sim$BUGSoutput$sims.matrix[,2]) #SMP21
```

```
## [1] 5.312125
```

```
mean(Smp1.sim$BUGSoutput$sims.matrix[,2]) #SMP1
```

```
## [1] 1.751189
```

Question 3

Analyze the LCD projector lamp failure time data in Table 4.2 using the gamma failure time model. Choose hyperparameters for the α and λ prior distributions so that the prior predictive distribution is similar to that for the Weibull failure time model used in Example 4.4. Evaluate the posterior distribution of the reliability function R(t) at 1,000 hours. Assess how well the gamma distribution fits these data. Compute DIC and compare with DIC calculated for the Weibull and lognormal distribution fits in Examples 4.4 and 4.5.

```
lcd.fail <- c(387, 182, 244, 600, 627, 332, 418, 300, 798, 584, 660, 39, 274, 174, 50, 34,
                  1895, 158, 974, 345, 1755, 1752, 473, 81, 954, 1407, 230, 464, 380, 131, 1205)


LCD_model <- "model {
  for (i in 1:31) {
  lcd.fail[i] ~ dgamma(alpha, lambda);
  }

  alpha ~ dgamma(2, .005)
  lambda ~ dgamma(.5, .001)

  PPD ~ dgamma(alpha, lambda)

}"

LCD.sim <- jags( data = c('lcd.fail'), parameters.to.save = c('alpha', 'lambda', 'PPD'),
  model.file = textConnection(LCD_model),
  n.iter = 22000,
  n.burnin=2000,
  n.chains=5,
  n.thin=1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 31
##    Unobserved stochastic nodes: 3
##    Total graph size: 38
##
## Initializing model
```

```
#head(LCD.sim$BUGSoutput$sims.matrix)

mean(LCD.sim$BUGSoutput$sims.matrix[,2] / LCD.sim$BUGSoutput$sims.matrix[,4]) # MEAN
```

```
## [1] 584.784
```

```
median(LCD.sim$BUGSoutput$sims.matrix[,2] / LCD.sim$BUGSoutput$sims.matrix[,4]) # MEDIAN
```

```
## [1] 575.2135
```

```
alpha <- LCD.sim$BUGSoutput$sims.matrix[,2]
lambda <- LCD.sim$BUGSoutput$sims.matrix[,4]

median(alpha) #median alpha
```

```
## [1] 1.337062
```

```
median(lambda) #median lambda
```

```
## [1] 0.002332506
```

```
mean(exp(-alpha*((1000)^lambda))) #evalute R(t) at 1,000 hours
```

```
## [1] 0.2621899
```

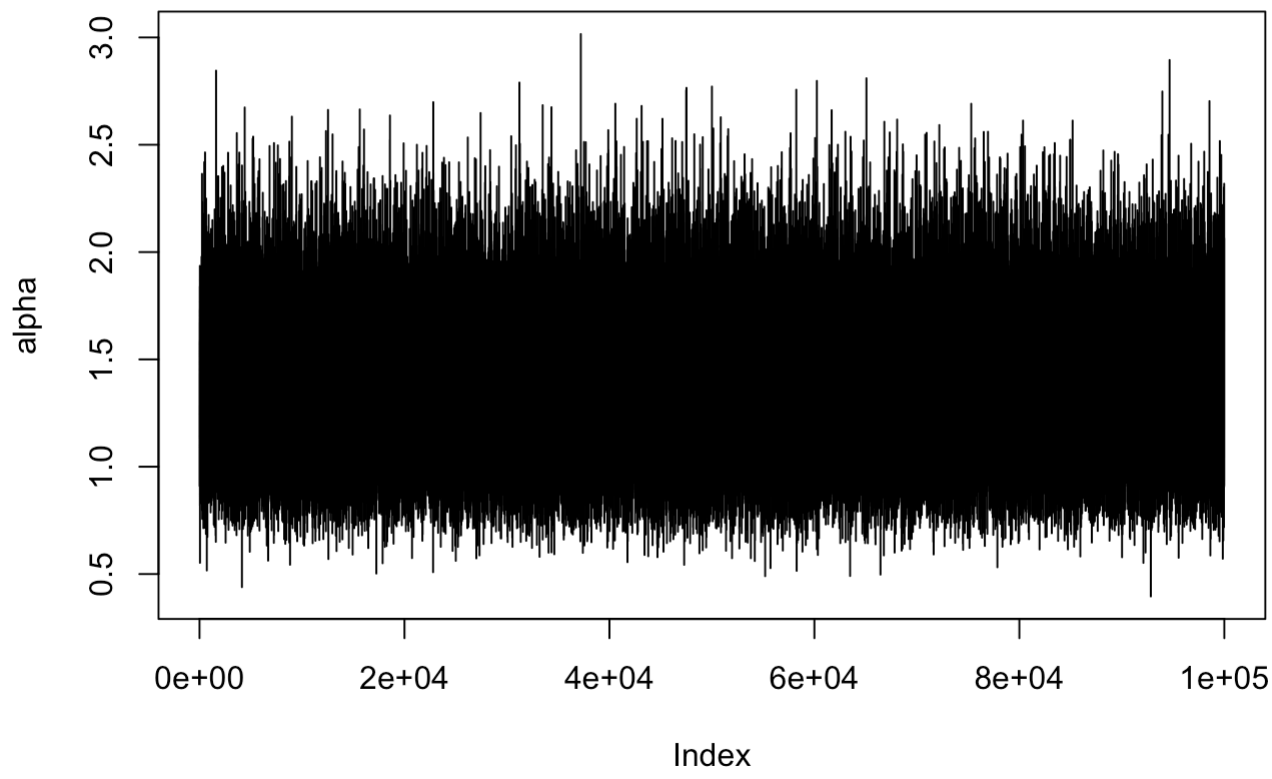
```
gammainc(mean(alpha), mean(lambda)*1000)
```

```
##    lowinc    uppinc    reginc
## 0.3510138 0.8765294 0.2859482
```

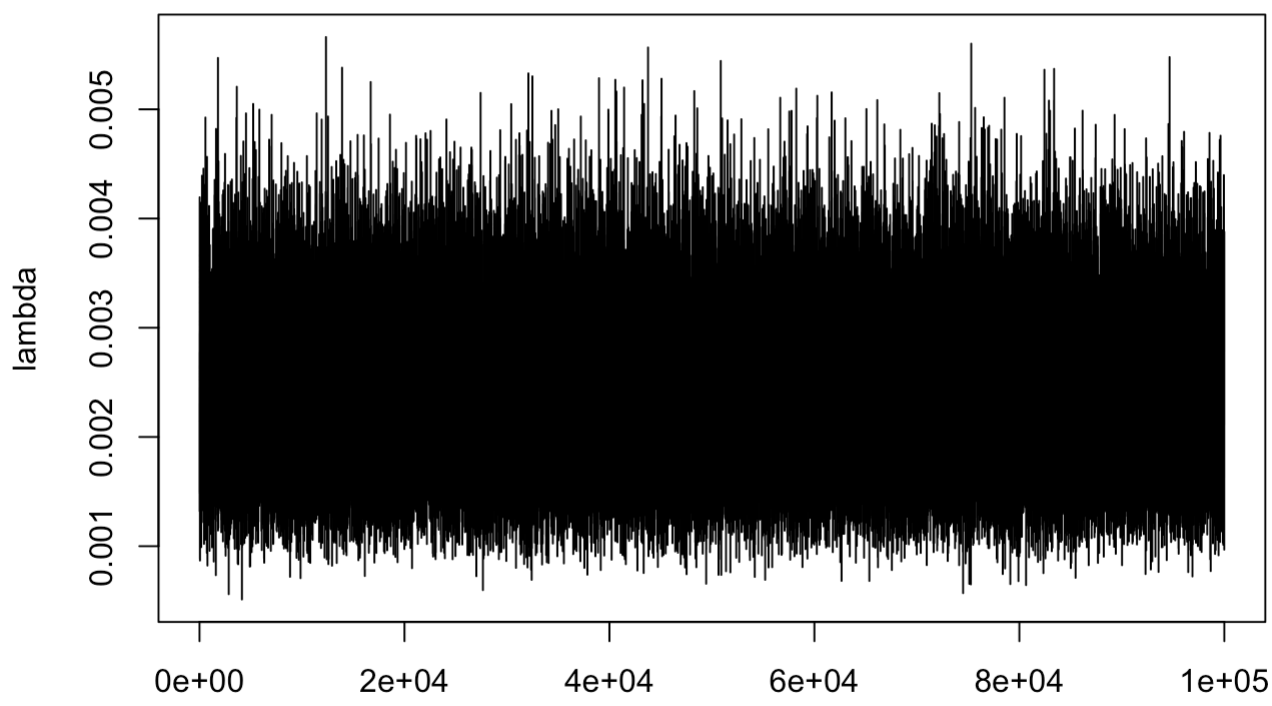
```
incgam(mean(alpha), mean(lambda)*1000) #evalute R(t) at 1,000 hours
```

```
## [1] 0.8765294
```

```
plot(alpha, type='l')
```




```
plot(lambda, type='l')
```




```
gelman.diag(LCD.sim$BUGSoutput,multivariate=F)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## alpha             1          1
## deviance          1          1
## lambda            1          1
## PPD               1          1
```

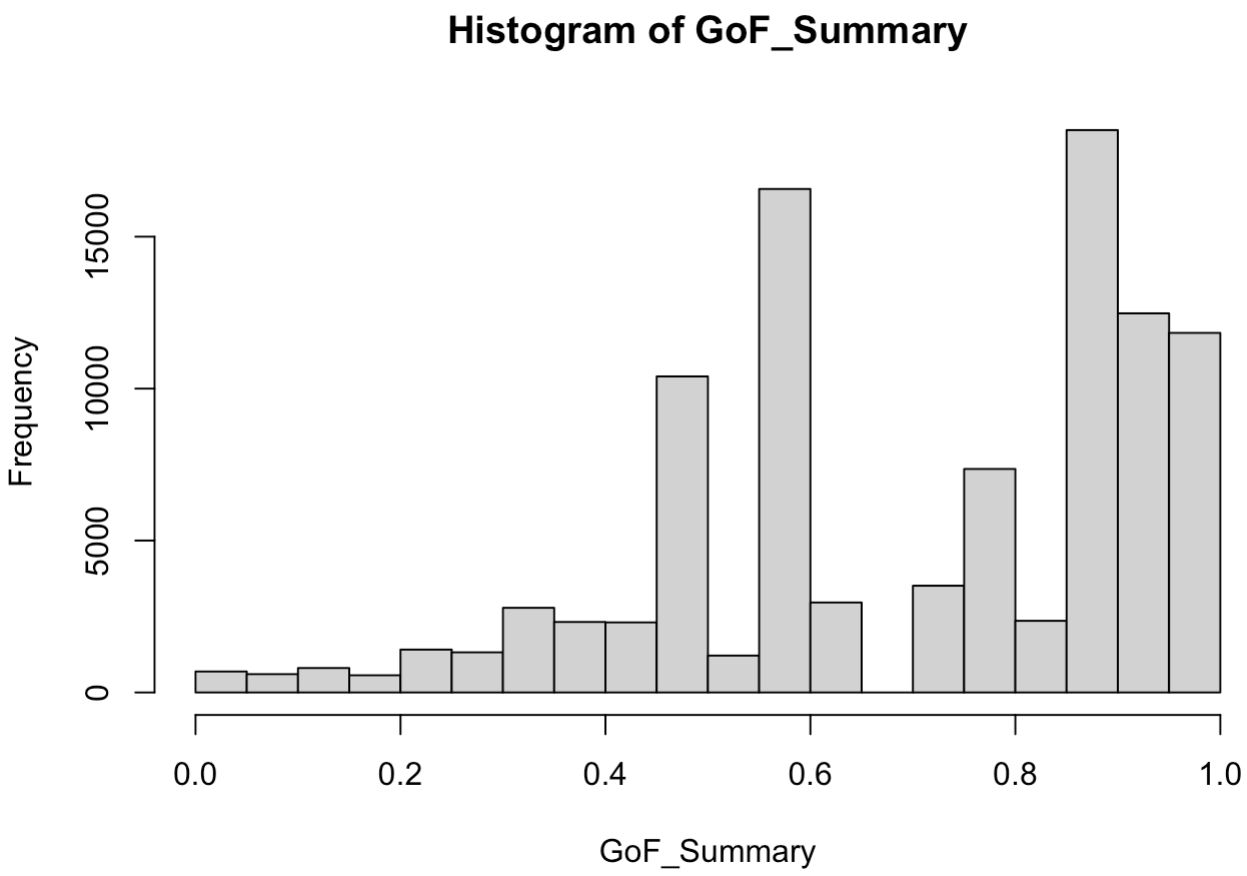
```
LCD.sim$BUGSoutput$DIC
```

```
## [1] 459.8086
```

```
GGchains <- LCD.sim$BUGSoutput$sims.matrix
effectiveSize(LCD.sim)
```

```
##     alpha  deviance    lambda       PPD
##  15355.36  28987.69  16905.21 100000.00
```

```
# Goodness of Fit
GoF <- matrix(NA,ncol=length(lcd.fail),nrow=length(GGchains[,3]))
for (i in 1:length(GGchains[,3])) {
  GoF[i,] <- pgamma(lcd.fail, alpha[i], lambda[i])
}

# Function requires fitted quantiles and returns a p-value
GoF_Test <- function(fitted_quantiles) {
  n <- length(fitted_quantiles)
  K <- round((n)^(0.4))
  mK <- table(cut(fitted_quantiles,(0:K)/K))
  np <- n/K
  RB <- sum(((mK-np)^2)/np)
  return(1-pchisq(RB,K-1))
}

# Calculating the p-values for each posterior model
GoF_Summary <- apply(GoF,1,GoF_Test)

# Histogram of posterior model p-values
hist(GoF_Summary,xlim=c(0,1))
```




```
# Percent of posterior models with p-value less than 0.05
mean(GoF_Summary < 0.05)
```

```
## [1] 0.0069
```

According to the Goodness-of-Fit test, the model fits the data very well. The gamma distribution model fits the data better than both examples from the textbook.

Question 4

Consider the following failure times (in 1,000 hours) for a particular component of an anti-aircraft missile system: 14.4, 2.1, 0.4, 18.6, 1.2, 2.6, 11.5, 18.4, 14.0, 2.8, 7.6, 2.7, 35.4, 10.4, 19.8, 11.3, 2.6, 0.8, 11.3, 5.4. Using the BIC model selection method in Sect. 4.6, determine which distribution among the exponential, Weibull, lognormal, or gamma distributions best fits the data. Assess the goodness of fit for these distributions using a Bayesian χ2 goodness-of-fit test.

```
missle <- c(14.4, 2.1, 0.4, 18.6, 1.2, 2.6, 11.5, 18.4, 14.0, 2.8, 7.6, 2.7, 35.4, 10.4, 19.8, 11.3, 2.6, 0.8, 1
1.3, 5.4)

# Exponential Model
MissleEXP <- " model {
  for (i in 1:length(missle)) {
    missle[i] ~ dexp(lambda)
  }
  lambda ~ dgamma(0.001, 0.001)
}
"

MissleEXP.sim <- jags(
  data = list(missle = missle),
  parameters.to.save = 'lambda',
  model.file = textConnection(MissleEXP),
  n.iter = 22000,
  n.burnin = 2000,
  n.chains = 5,
  n.thin = 1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 20
##    Unobserved stochastic nodes: 1
##    Total graph size: 22
##
## Initializing model
```

```
#head(MissleEXP.sim$BUGSoutput$sims.matrix)
lambda <- MissleEXP.sim$BUGSoutput$sims.matrix[,2]
gelman.diag(MissleEXP.sim$BUGSoutput,multivariate=F)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## deviance          1          1
## lambda            1          1
```

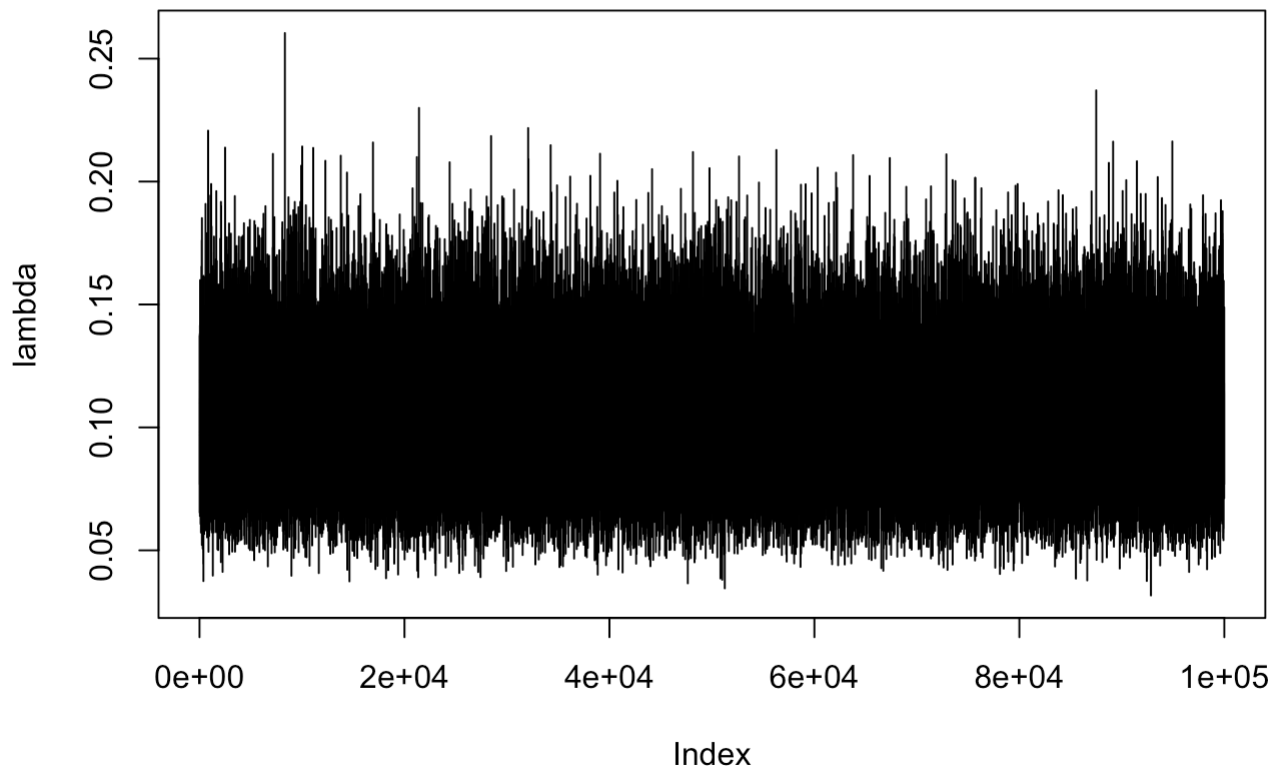
```
EXPchains <- MissleEXP.sim$BUGSoutput$sims.matrix
effectiveSize(MissleEXP.sim)
```

```
##  deviance    lambda
##  98687.13 100000.00
```

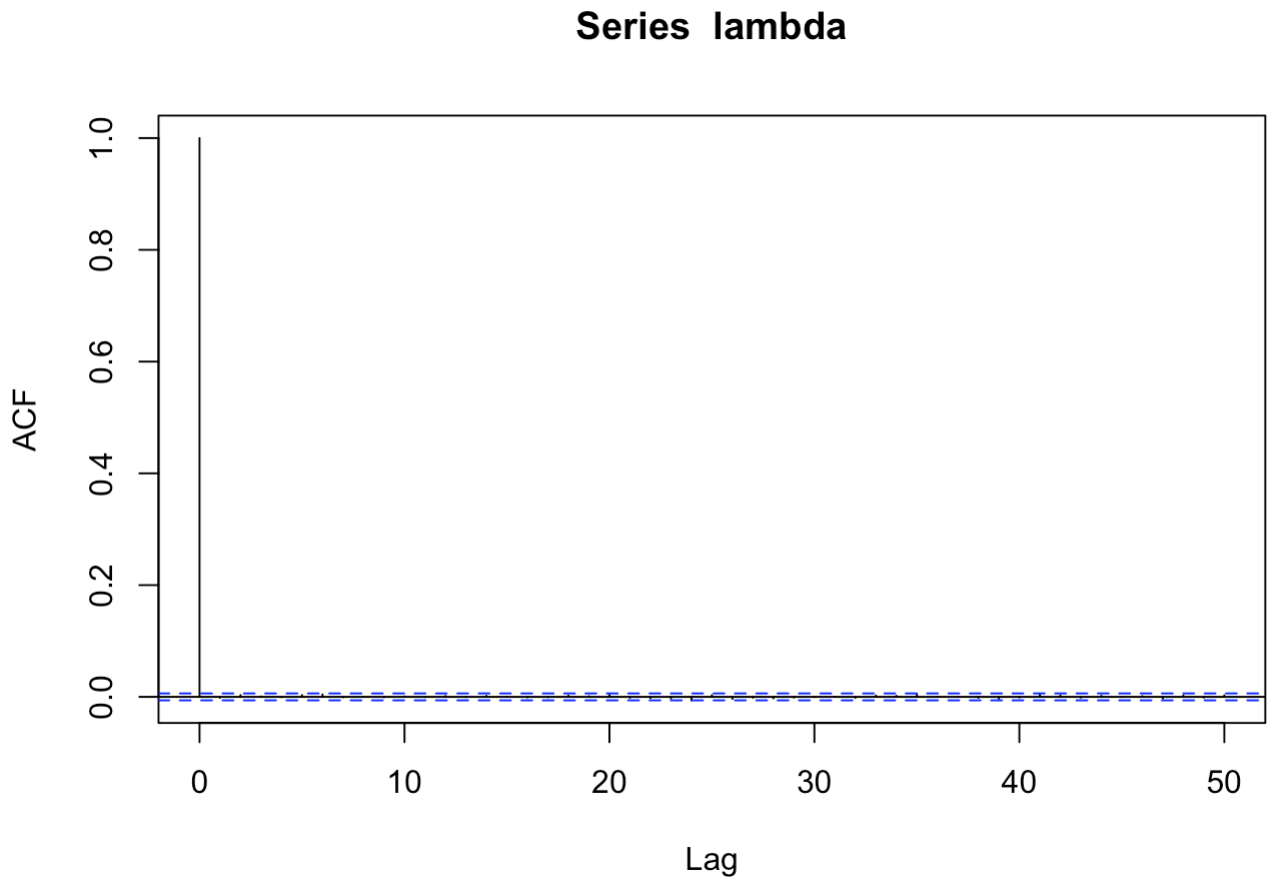
```
MissleEXP.sim$BUGSoutput$DIC
```

```
## [1] 132.7773
```

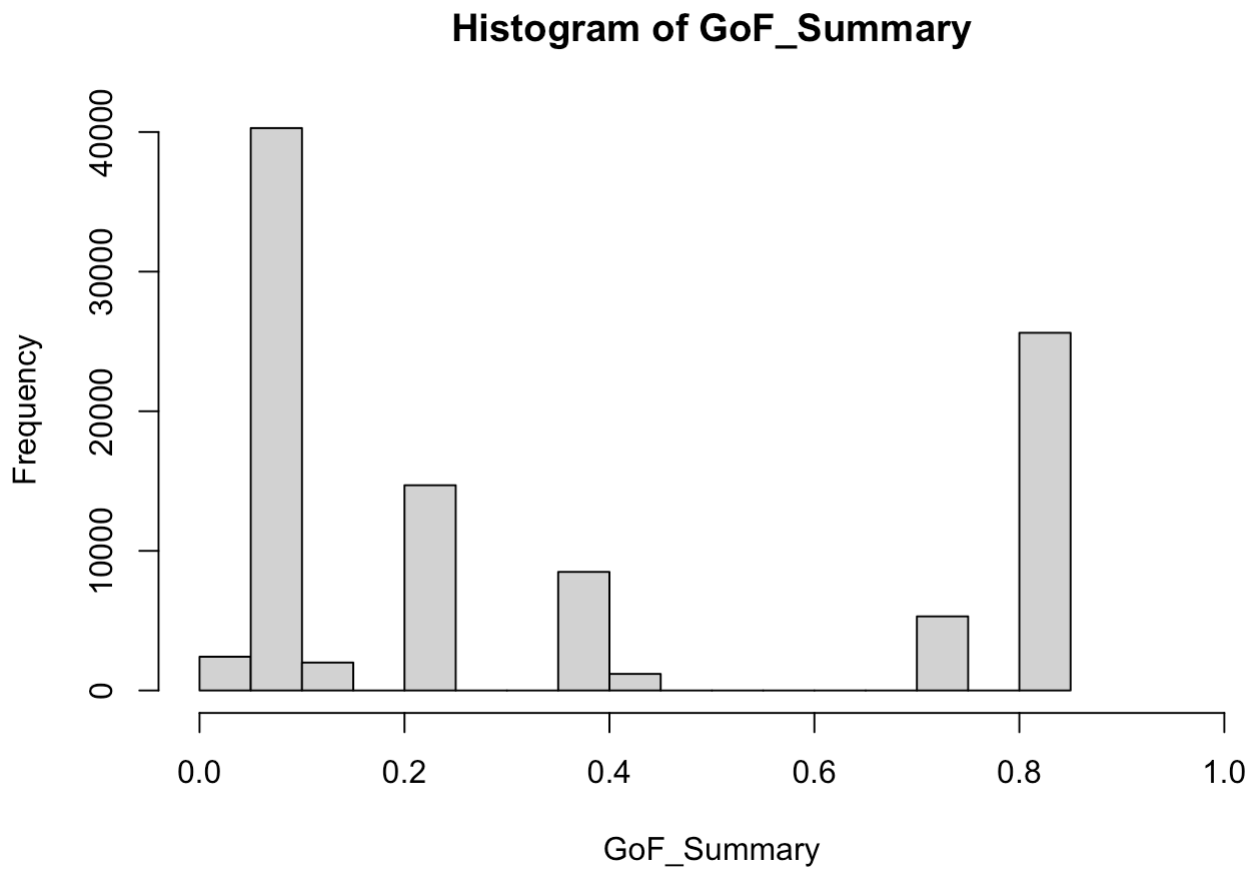
```
plot(lambda, type='l')
```




```
acf(lambda)
```




```
# Goodness of Fit
GoF <- matrix(NA,ncol=length(missle),nrow=length(EXPchains[,1]))
for (i in 1:length(EXPchains[,1])) {
  GoF[i,] <- pexp(missle, lambda[i])
}

# Function requires fitted quantiles and returns a p-value
GoF_Test <- function(fitted_quantiles) {
  n <- length(fitted_quantiles)
  K <- round((n)^(0.4))
  mK <- table(cut(fitted_quantiles,(0:K)/K))
  np <- n/K
  RB <- sum(((mK-np)^2)/np)
  return(1-pchisq(RB,K-1))
}

# Calculating the p-values for each posterior model
GoF_Summary <- apply(GoF,1,GoF_Test)

# Histogram of posterior model p-values
hist(GoF_Summary,xlim=c(0,1))
```




```
# Percent of posterior models with p-value less than 0.05
mean(GoF_Summary < 0.05)
```

```
## [1] 0.02416
```

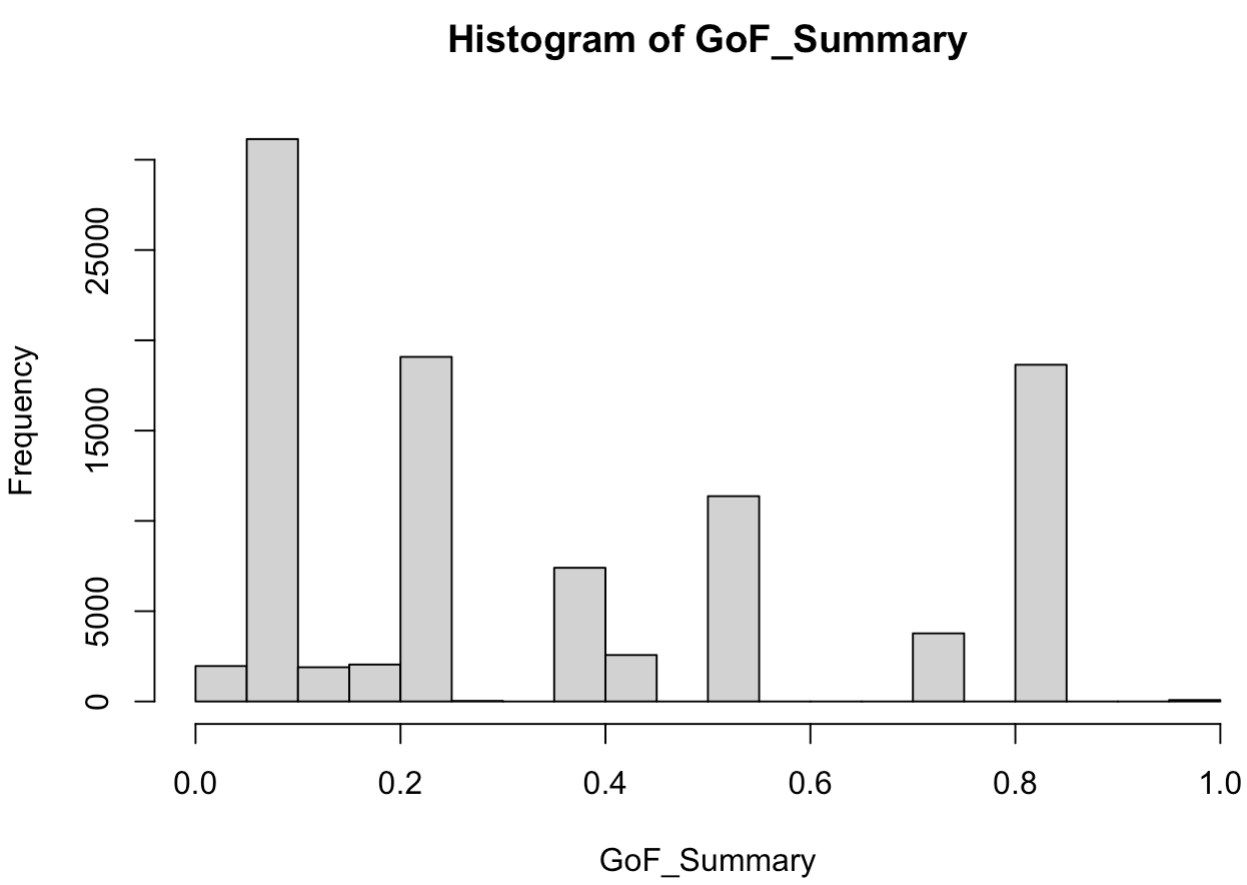
```
#mean(1/lambda)


# Weibull Model
MissleWEIB <- " model {
  for (i in 1:length(missle)) {
    missle[i] ~ dweib(v, lambda)
  }

  v ~ dgamma(0.001, 0.001)
  lambda ~ dgamma(0.001, 0.001)
}
"

MissleWEIB.sim <- jags(
  data = list(missle = missle),
  parameters.to.save = c('v', 'lambda'),
  model.file = textConnection(MissleWEIB),
  n.iter = 22000,
  n.burnin = 2000,
  n.chains = 5,
  n.thin = 1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 20
##    Unobserved stochastic nodes: 2
##    Total graph size: 23
##
## Initializing model
```

```
#head(MissleWEIB.sim$BUGSoutput$sims.matrix)
a <- MissleWEIB.sim$BUGSoutput$sims.matrix[,3]
b <- (MissleWEIB.sim$BUGSoutput$sims.matrix[,2])^(-1/a)
#plot(a, type='l')
#plot(b, type='l')
#gelman.diag(MissleWEIB.sim$BUGSoutput,multivariate=F)
WEIBchains <- MissleWEIB.sim$BUGSoutput$sims.matrix
#effectiveSize(MissleWEIB.sim)
MissleWEIB.sim$BUGSoutput$DIC
```

```
## [1] 134.702
```

```
# Goodness of Fit
GoF <- matrix(NA,ncol=length(missle),nrow=length(WEIBchains[,1]))
for (i in 1:length(WEIBchains[,1])) {
  GoF[i,] <- pweibull(missle, a[i], b[i])
}

# Function requires fitted quantiles and returns a p-value
GoF_Test <- function(fitted_quantiles) {
  n <- length(fitted_quantiles)
  K <- round((n)^(0.4))
  mK <- table(cut(fitted_quantiles,(0:K)/K))
  np <- n/K
  RB <- sum(((mK-np)^2)/np)
  return(1-pchisq(RB,K-1))
}

# Calculating the p-values for each posterior model
GoF_Summary <- apply(GoF,1,GoF_Test)

# Histogram of posterior model p-values
hist(GoF_Summary,xlim=c(0,1))
```




```
# Percent of posterior models with p-value less than 0.05
mean(GoF_Summary < 0.05)
```

```
## [1] 0.01962
```

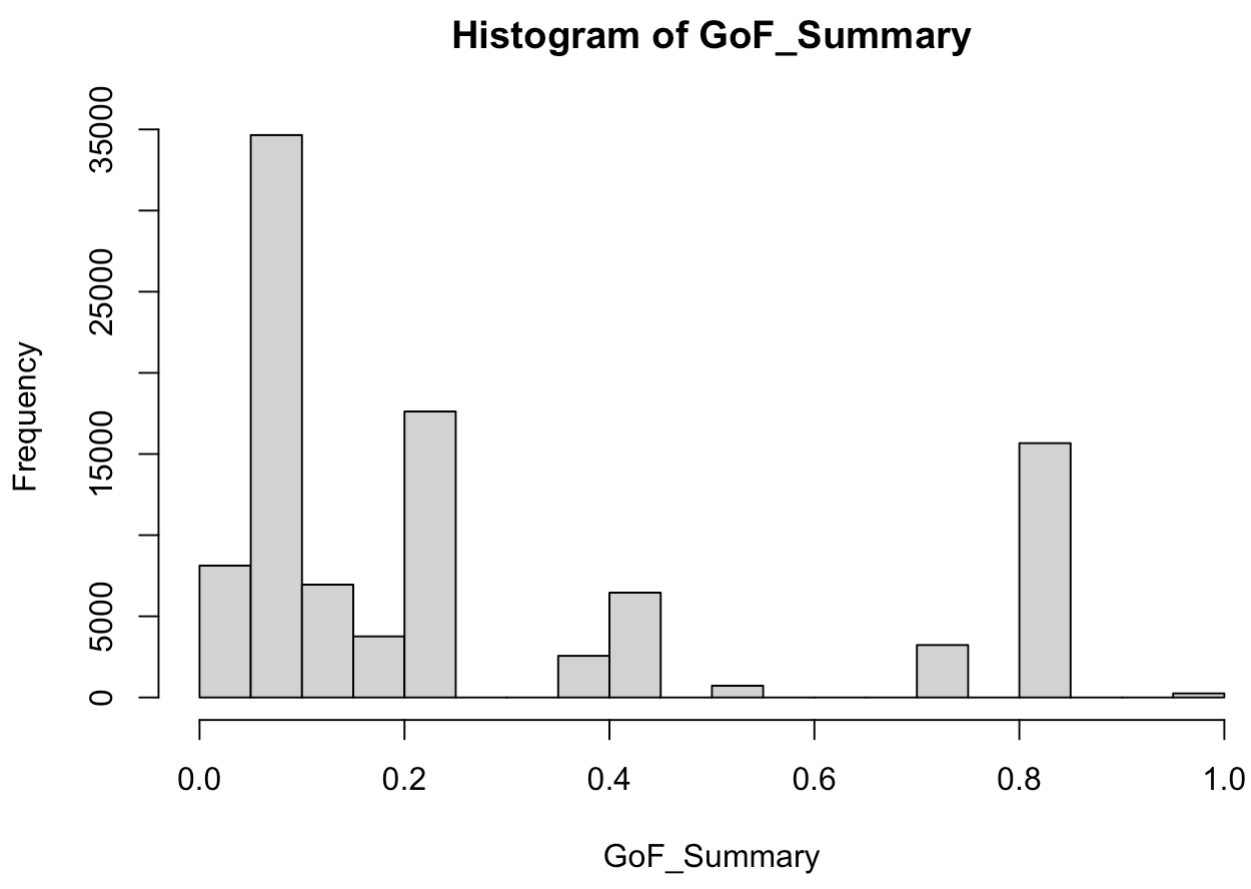
```
#Log-Normal Model
lmissle <- log(missle)

MissleLN <- "model {
  for (i in 1:length(lmissle)) {
    lmissle[i] ~ dnorm(mu, 1/(sigma^2))
  }
  mu ~ dnorm(0, 1/1000)
  sigma ~ dexp(1/1000)
}
"

MissleLN.sim <- jags(
  data = list(lmissle = lmissle),
  parameters.to.save = c('mu', 'sigma'),
  model.file = textConnection(MissleLN),
  n.iter = 22000,
  n.burnin = 2000,
  n.chains = 5,
  n.thin = 1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 20
##    Unobserved stochastic nodes: 2
##    Total graph size: 29
##
## Initializing model
```

```
#head(MissleLN.sim$BUGSoutput$sims.matrix)
mu <- MissleLN.sim$BUGSoutput$sims.matrix[,2]
sigma <- MissleLN.sim$BUGSoutput$sims.matrix[,3]
#plot(mu, type='l')
#plot(sigma, type='l')
#gelman.diag(MissleLN.sim$BUGSoutput,multivariate=F)
LNchains <- MissleLN.sim$BUGSoutput$sims.matrix
#effectiveSize(MissleLN.sim)
MissleLN.sim$BUGSoutput$DIC
```

```
## [1] 68.10208
```

```
# Goodness of Fit
GoF <- matrix(NA,ncol=length(missle),nrow=length(LNchains[,3]))
for (i in 1:length(LNchains[,3])) {
  GoF[i,] <- pnorm(lmissle, mu[i], sigma[i])
}

# Function requires fitted quantiles and returns a p-value
GoF_Test <- function(fitted_quantiles) {
  n <- length(fitted_quantiles)
  K <- round((n)^(0.4))
  mK <- table(cut(fitted_quantiles,(0:K)/K))
  np <- n/K
  RB <- sum(((mK-np)^2)/np)
  return(1-pchisq(RB,K-1))
}

# Calculating the p-values for each posterior model
GoF_Summary <- apply(GoF,1,GoF_Test)

# Histogram of posterior model p-values
hist(GoF_Summary,xlim=c(0,1))
```




```
# Percent of posterior models with p-value less than 0.05
mean(GoF_Summary < 0.05)
```

```
## [1] 0.08123
```

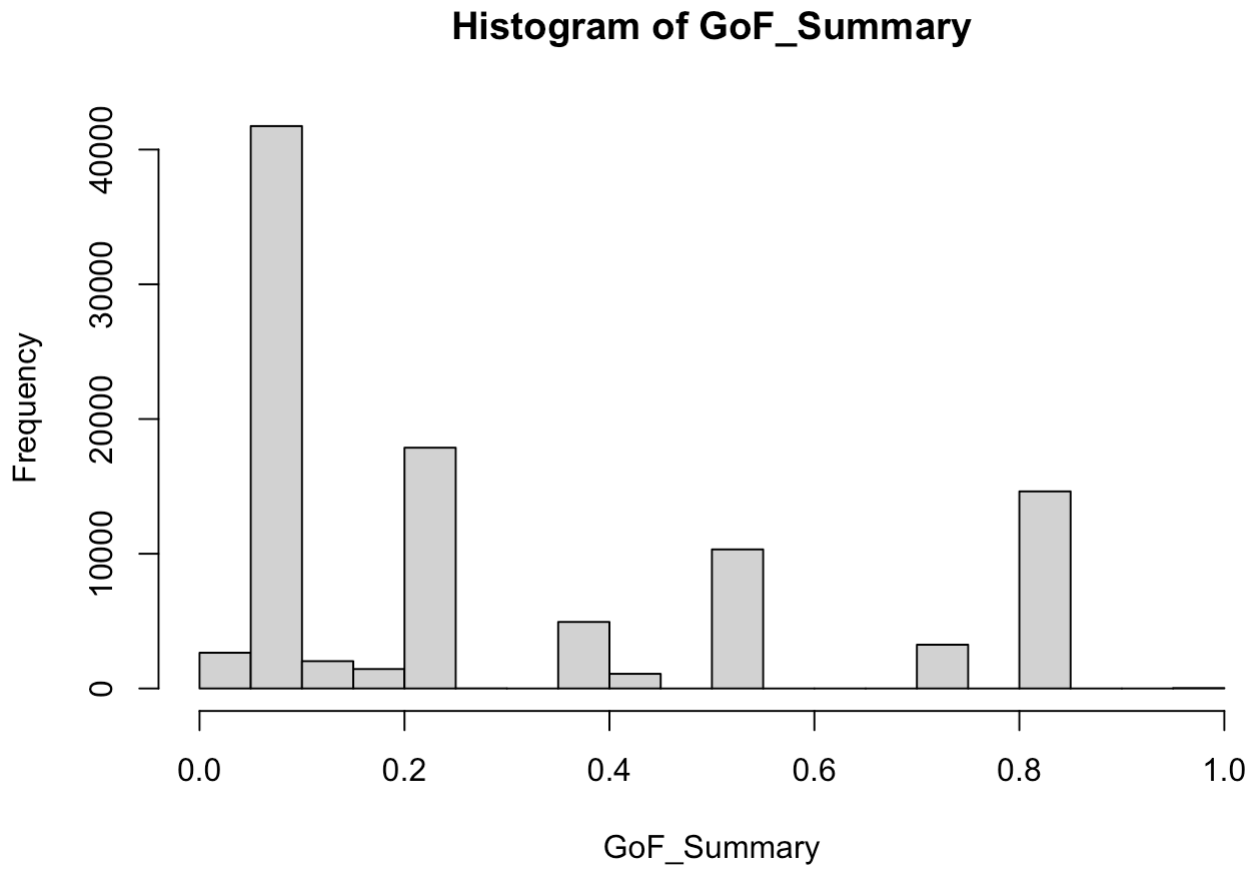
```
# Gamma Model
MissleGamma <- "model {
  for (i in 1:length(missle)) {
    missle[i] ~ dgamma(alpha, beta)
  }
  beta ~ dgamma(1, .1)
  alpha ~ dgamma(1, .1)
}
"

MissleGamma.sim <- jags(
  data = list(missle = missle),
  parameters.to.save = c('alpha', 'beta'),
  model.file = textConnection(MissleGamma),
  n.iter = 22000,
  n.burnin = 2000,
  n.chains = 5,
  n.thin = 1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 20
##    Unobserved stochastic nodes: 2
##    Total graph size: 24
##
## Initializing model
```

```
#head(MissleGamma.sim$BUGSoutput$sims.matrix)
alpha <- MissleGamma.sim$BUGSoutput$sims.matrix[,1]
beta <- MissleGamma.sim$BUGSoutput$sims.matrix[,2]
#plot(alpha, type='l')
#plot(beta, type='l')
#acf(alpha)
#acf(beta)
#gelman.diag(MissleGamma.sim$BUGSoutput,multivariate=F)
Gammachains <- MissleGamma.sim$BUGSoutput$sims.matrix
#effectiveSize(MissleGamma.sim)
MissleGamma.sim$BUGSoutput$DIC
```

```
## [1] 134.8334
```

```
# Goodness of Fit
GoF <- matrix(NA,ncol=length(missle),nrow=length(Gammachains[,3]))
for (i in 1:length(Gammachains[,3])) {
  GoF[i,] <- pgamma(missle, shape=alpha[i], rate=beta[i])
}

# Function requires fitted quantiles and returns a p-value
GoF_Test <- function(fitted_quantiles) {
  n <- length(fitted_quantiles)
  K <- round((n)^(0.4))
  mK <- table(cut(fitted_quantiles,(0:K)/K))
  np <- n/K
  RB <- sum(((mK-np)^2)/np)
  return(1-pchisq(RB,K-1))
}

# Calculating the p-values for each posterior model
GoF_Summary <- apply(GoF,1,GoF_Test)

# Histogram of posterior model p-values
hist(GoF_Summary,xlim=c(0,1))
```




```
# Percent of posterior models with p-value less than 0.05
mean(GoF_Summary < 0.05)
```

```
## [1] 0.0265
```

```
#mean(alpha/beta)
```

Log-Normal model has the lowest DIC and fits the data well; however, the Weibull model seems to fit the data the best, as it has the lowest percent of posterior models with a p-value less than 0.05.

Question 5

Using the DIC model selection method in Sect. 4.6, determine which distribution among the exponential, Weibull, lognormal, or gamma distributions best fits the data. Assess the goodness of fit for these distributions using a Bayesian χ2 goodness-of-fit test.

```
vacuum <- read.csv('table48.txt', sep="", header=TRUE)
#head(vacuum)
vac.fail <- vacuum$Failure
n <- length(vac.fail)

cens <- rep(0,n)
cens[1] <- 1   #index of which values are censored

cindex <- which(cens==0) #index of which values are censored

t <- vac.fail          #Failure times
t[cindex] <- NA        #NA for censoring since they didn't fail
c <- vac.fail          #Censoring times
I <- 1-cens

Vacuum.Exp <- "model {
  for (i in 1:123) {
    I[i] ~ dinterval(t[i], c[i])
    t[i] ~ dexp(lambda)
  }
  lambda ~ dgamma(0.001, 0.001)
}
"

Vacuum.Exp.sim <- jags(
  data = c('t', 'c', 'I'),
  parameters.to.save = c('lambda'),
  model.file = textConnection(Vacuum.Exp),
  n.iter = 22000,
  n.burnin = 2000,
  n.chains = 5,
  n.thin = 1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 124
##    Unobserved stochastic nodes: 123
##    Total graph size: 371
##
## Initializing model
```

```
#head(Vacuum.Exp.sim$BUGSoutput$sims.matrix)
length(Vacuum.Exp.sim$BUGSoutput$sims.matrix[,1])
```

```
## [1] 100000
```

```
lambda <- Vacuum.Exp.sim$BUGSoutput$sims.matrix[,2]

gelman.diag(Vacuum.Exp.sim$BUGSoutput,multivariate=F)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## deviance       1.07       1.17
## lambda         1.02       1.06
```

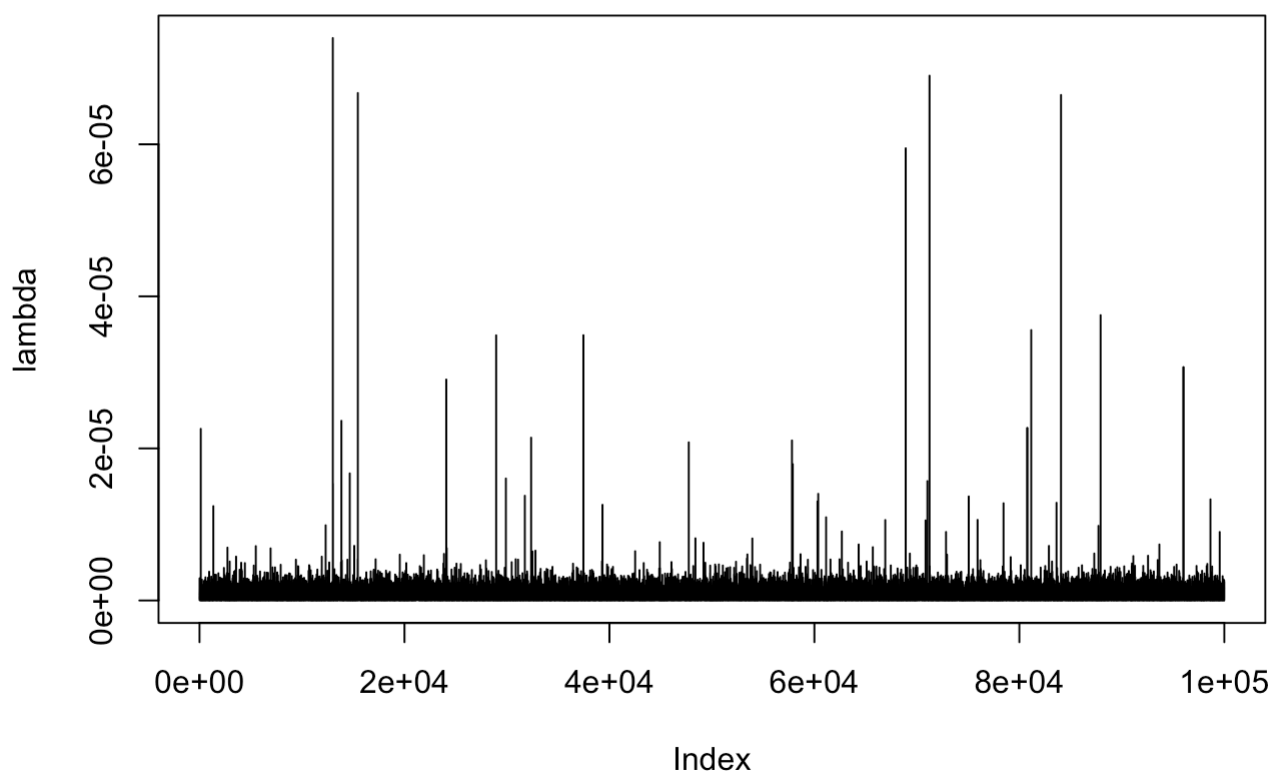
```
Vacuum.Exp.sim$BUGSoutput$DIC
```

```
## [1] 33.37106
```

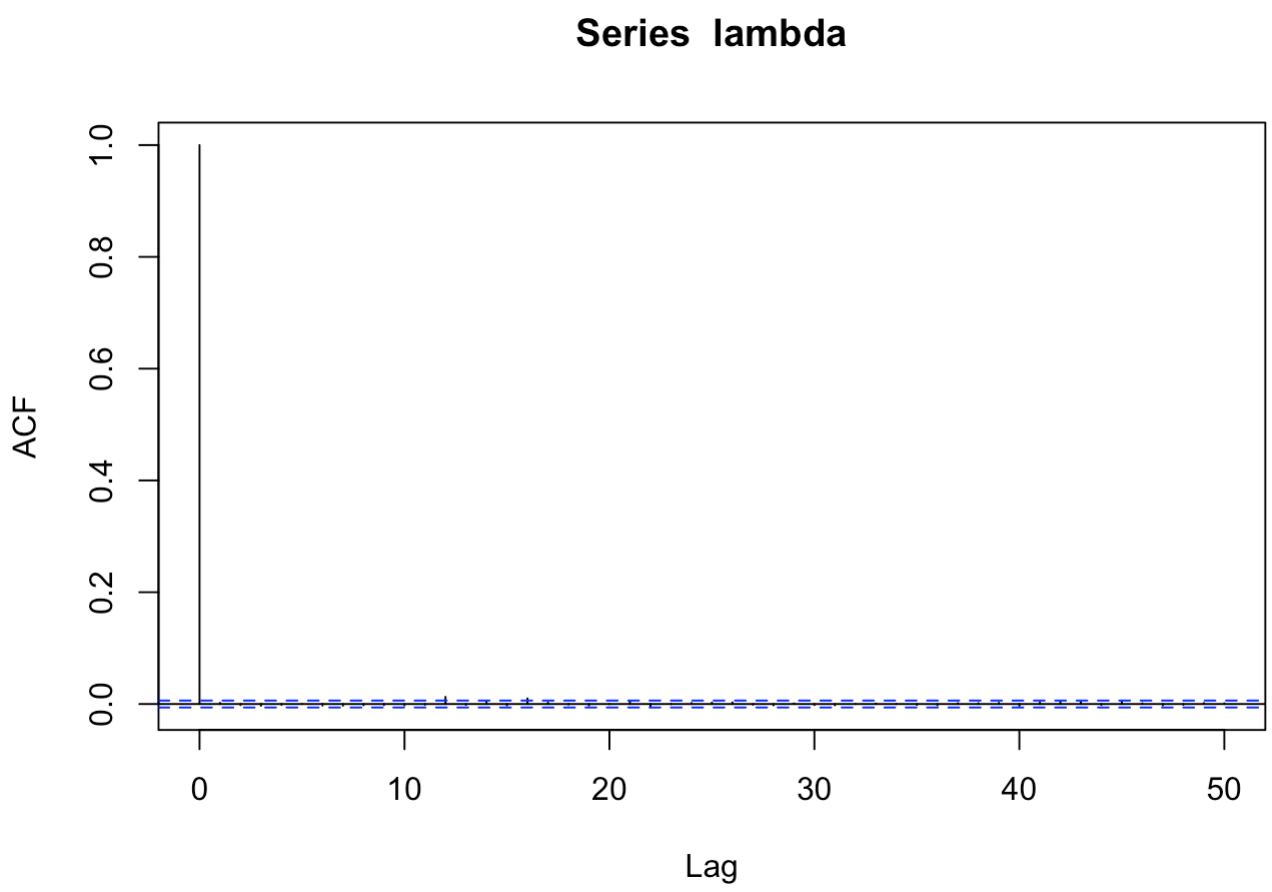
```
effectiveSize(Vacuum.Exp.sim)
```

```
##  deviance    lambda
##  222.5508 1309.2158
```

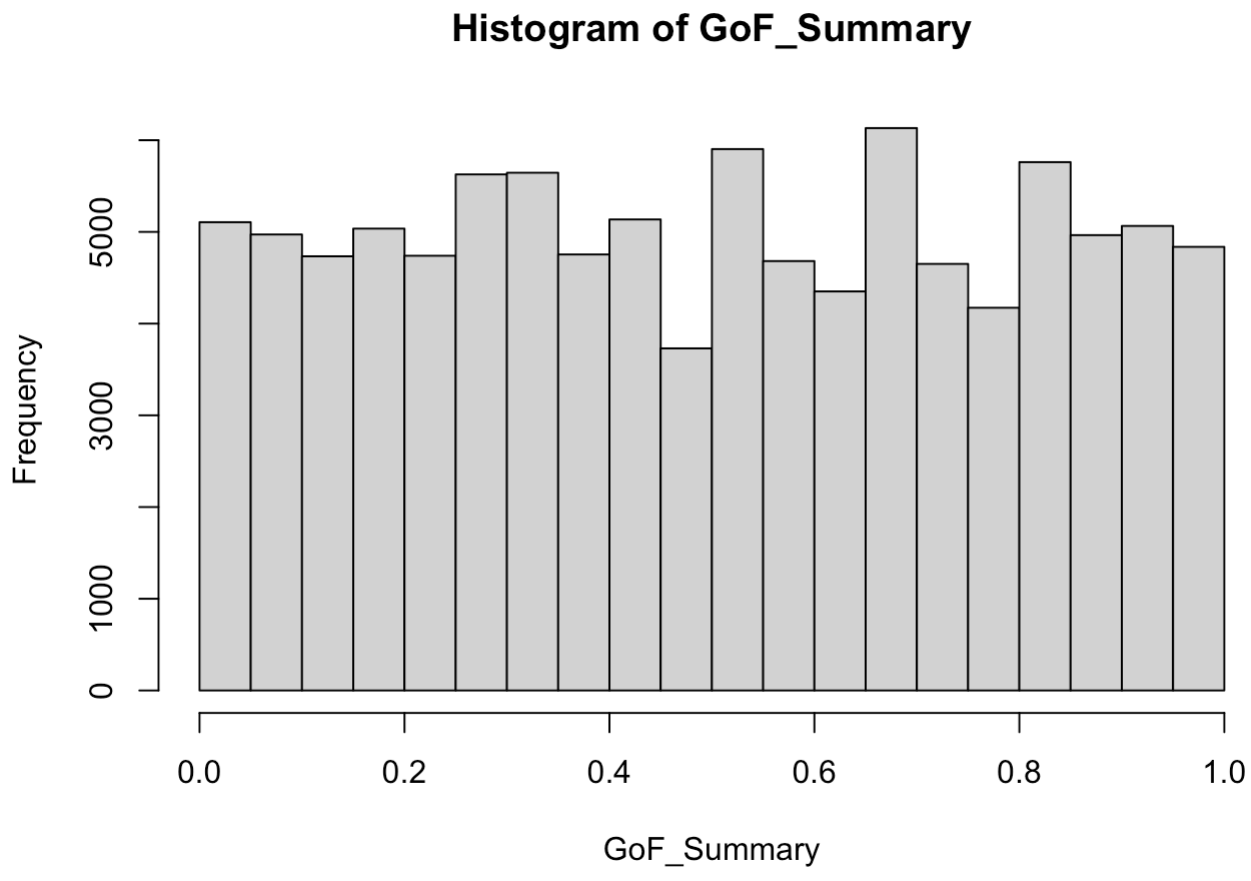
```
plot(lambda, type='l')
```




```
acf(lambda)
```




```
GoF <- matrix(NA,ncol=length(vac.fail),nrow=length(lambda))
for (i in 1:length(lambda)) {
  GoF[i,] <- pexp(vac.fail,lambda[i])
  temp <- runif(length(vac.fail),GoF[i,],1)
  GoF[i,] <- pmax(GoF[i,],temp*I)
}

# Function requires fitted quantiles and returns a p-value
GoF_Test <- function(fitted_quantiles) {
  n <- length(fitted_quantiles)
  K <- round((n)^(0.4))
  mK <- table(cut(fitted_quantiles,(0:K)/K))
  np <- n/K
  RB <- sum(((mK-np)^2)/np)
  return(1-pchisq(RB,K-1))
}

# Calculating the p-values for each posterior model
GoF_Summary <- apply(GoF,1,GoF_Test)

# Histogram of posterior model p-values
hist(GoF_Summary,xlim=c(0,1))
```




```
# Percent of posterior models with p-value less than 0.05
mean(GoF_Summary < 0.05)
```

```
## [1] 0.05106
```

```
Weibull_model <- "model {
  for (i in 1:123) {
  vac.fail[i] ~ dweib(v, lambda);
  }
  v ~ dgamma(.001, .001)
  lambda ~ dgamma(.001, .001)
}
"
Weibull_model.sim <- jags( data = c('vac.fail'), parameters.to.save = c('v', 'lambda'),
                   model.file = textConnection(Weibull_model),
                   n.iter = 22000,
                   n.burnin=2000,
                   n.chains=5,
                   n.thin=1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 123
##    Unobserved stochastic nodes: 2
##    Total graph size: 126
##
## Initializing model
```

```
#head(Weibull_model.sim$BUGSoutput$sims.matrix)

lambda <- Weibull_model.sim$BUGSoutput$sims.matrix[,2]
v <- Weibull_model.sim$BUGSoutput$sims.matrix[,3]

gelman.diag(Weibull_model.sim$BUGSoutput,multivariate=F)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## deviance       1.10       1.19
## lambda         1.30       1.79
## v              1.05       1.11
```

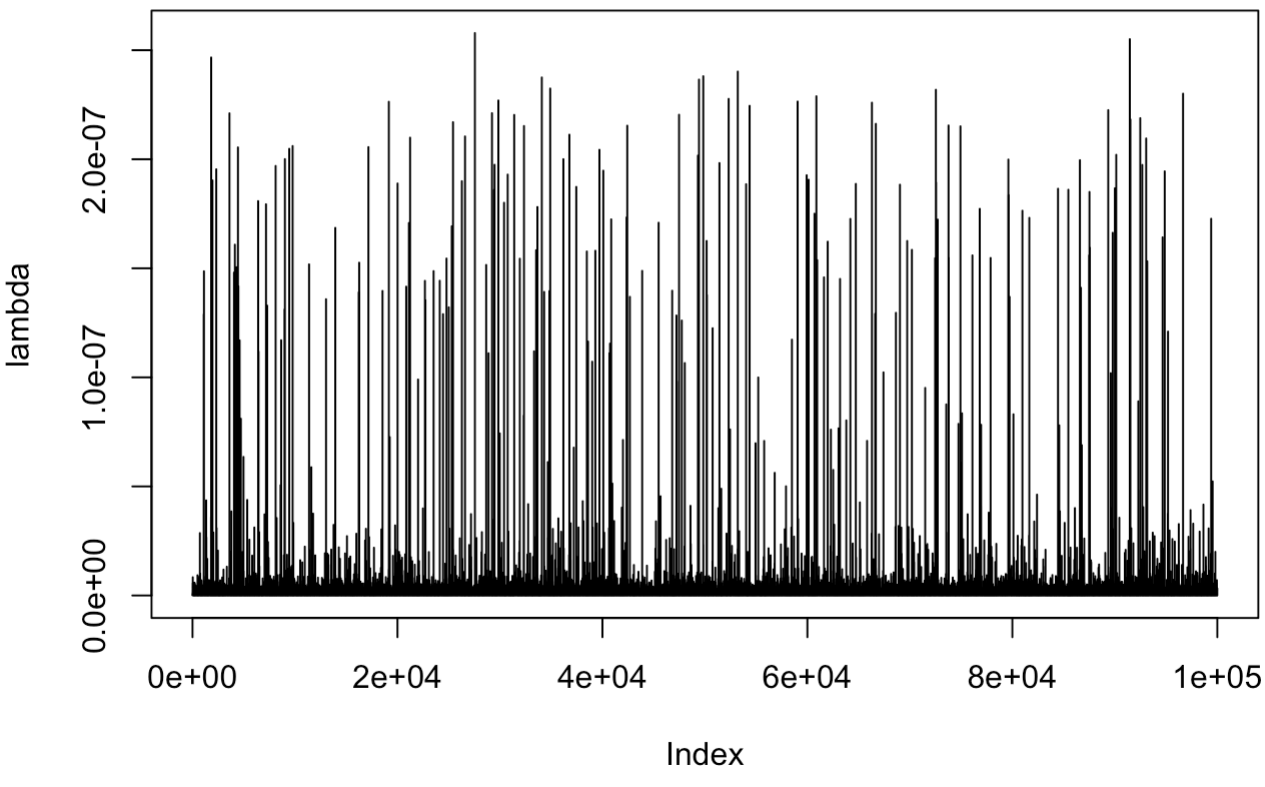
```
Weibull_model.sim$BUGSoutput$DIC
```

```
## [1] 2500.531
```

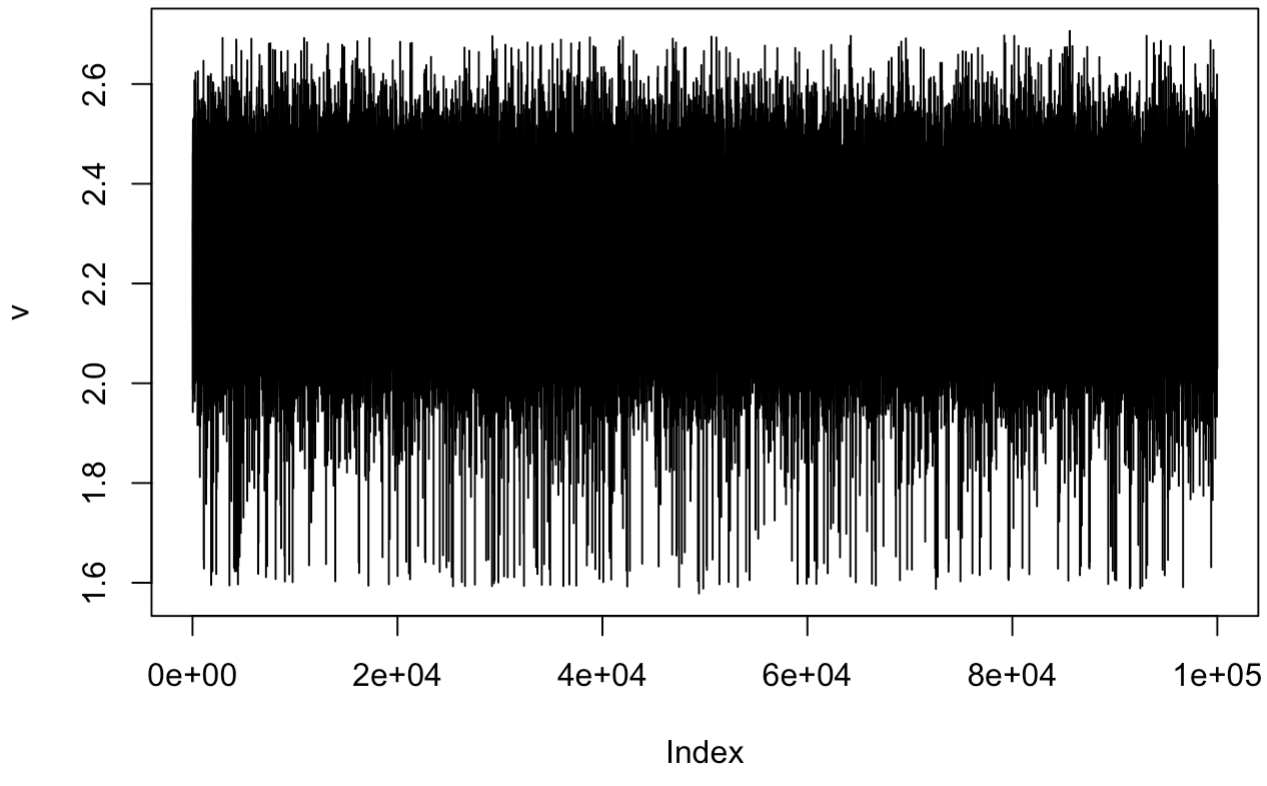
```
effectiveSize(Weibull_model.sim)
```

```
## deviance   lambda        v
## 503.9769   0.0000 179.7983
```

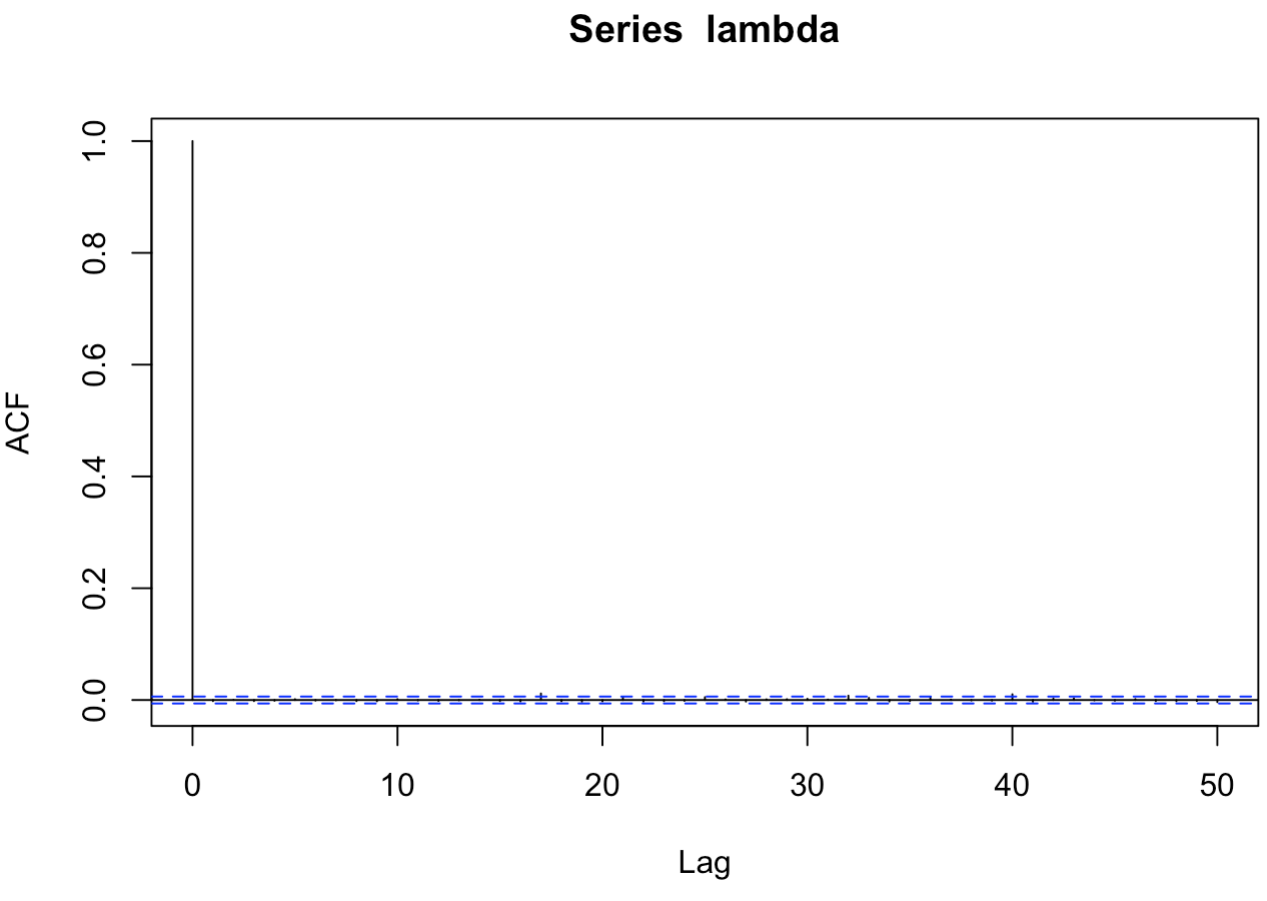
```
plot(lambda, type='l')
```



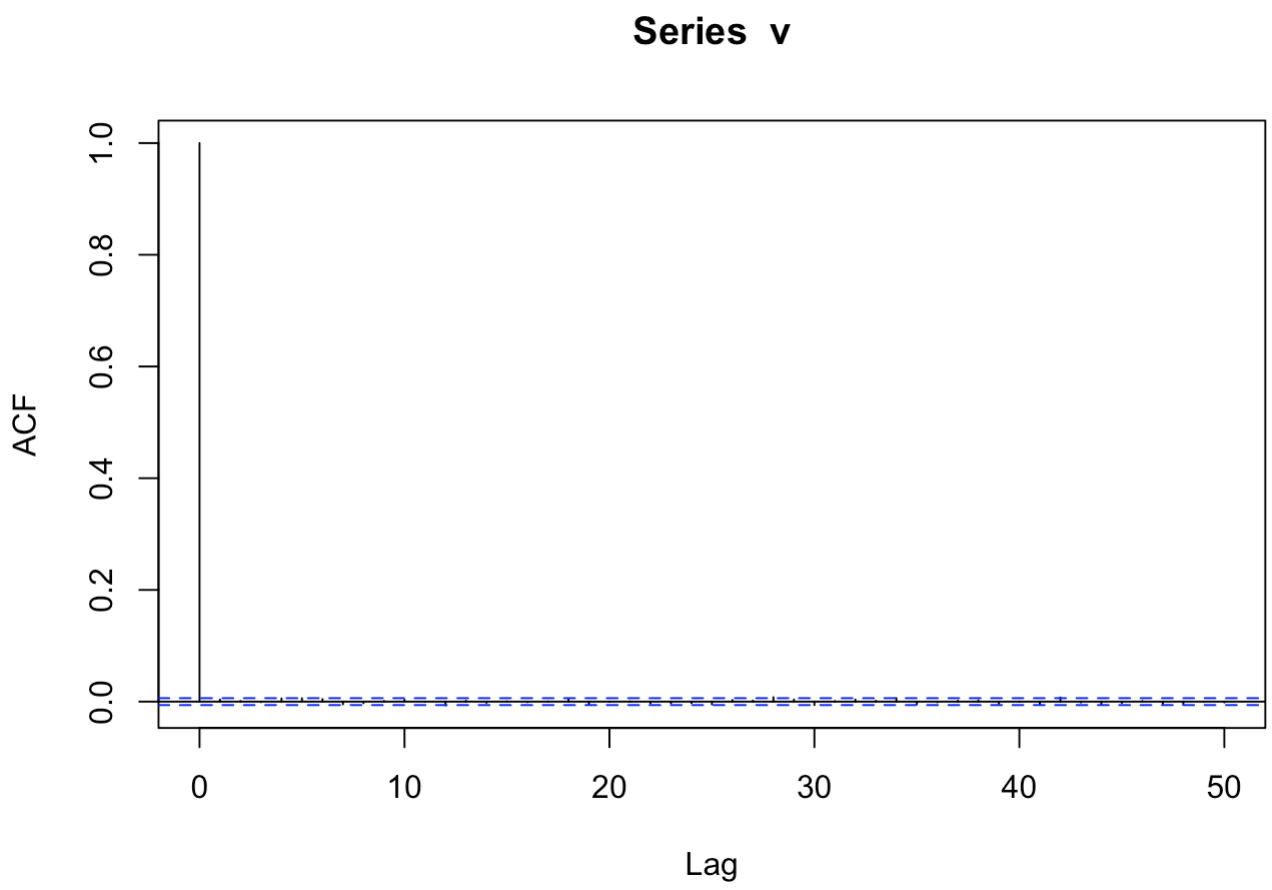
```
plot(v, type='l')
```



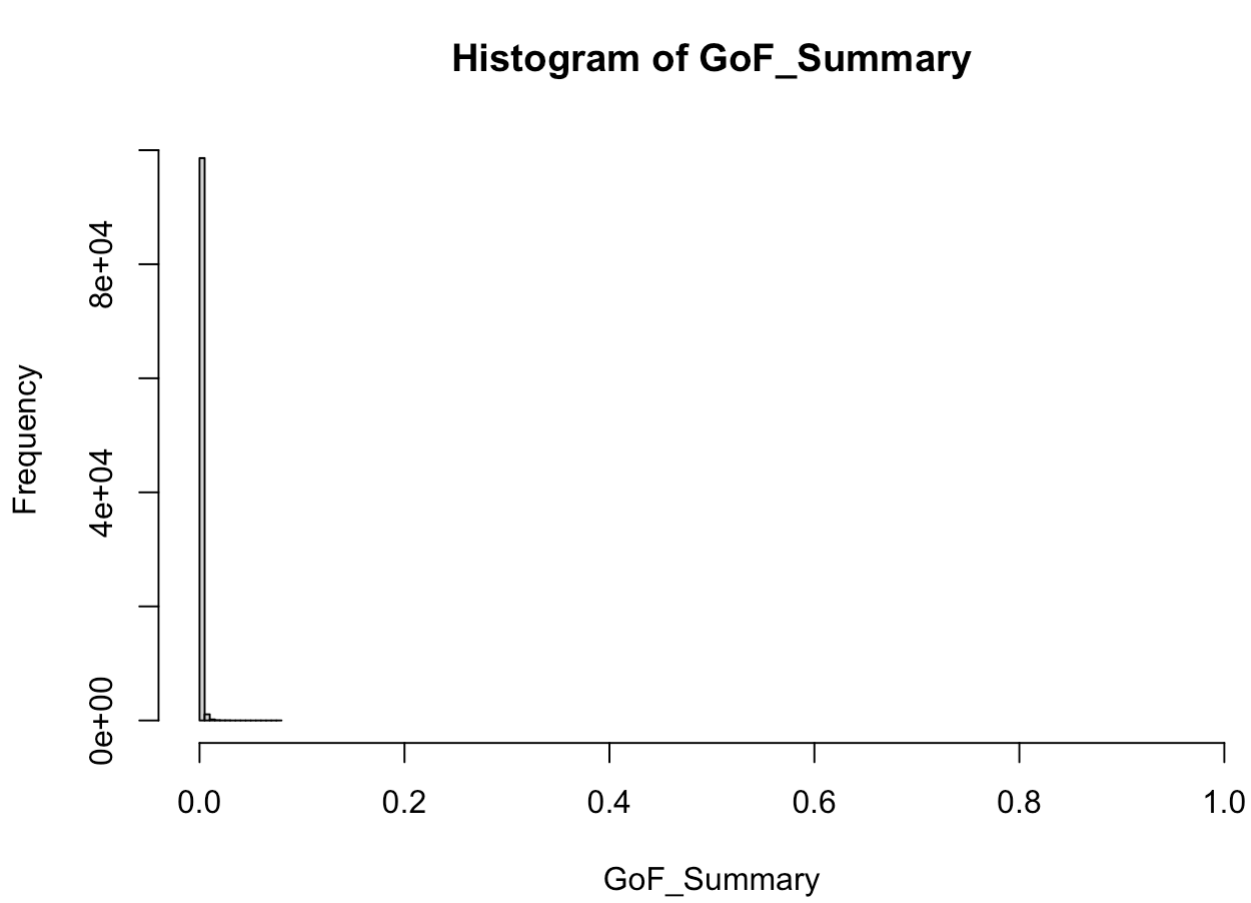

```
acf(lambda)
```




```
acf(v)
```




```
#chains <- as.mcmc(Weibull_model.sim$BUGSoutput$sims.matrix)
a <- v
b <- lambda^(-1/a)

GoF <- matrix(NA,ncol=length(vac.fail),nrow=length(a))
for (i in 1:length(a)) {
  GoF[i,] <- pweibull(vac.fail, a[i], b[i])
}

# Function requires fitted quantiles and returns a p-value
GoF_Test <- function(fitted_quantiles) {
  n <- length(fitted_quantiles)
  K <- round((n)^(0.4))
  mK <- table(cut(fitted_quantiles,(0:K)/K))
  np <- n/K
  RB <- sum(((mK-np)^2)/np)
  return(1-pchisq(RB,K-1))
}

# Calculating the p-values for each posterior model
GoF_Summary <- apply(GoF,1,GoF_Test)

# Histogram of posterior model p-values
hist(GoF_Summary,xlim=c(0,1))
```




```
# Percent of posterior models with p-value less than 0.05
mean(GoF_Summary < 0.05)
```

```
## [1] 0.99995
```

```
VacuumLogN <- "model {
  for(i in 1:123){
    I[i] ~ dinterval(t[i],c[i])
    t[i] ~ dlnorm(mu, tau)
  }
  mu ~ dnorm(0,.001)
  tau ~ dexp(.001)
}
"
VacuumLogN.sim <- jags(
  data=c('t','c','I'),
  parameters.to.save=c('mu','tau'),
  model.file=textConnection(VacuumLogN),
  n.iter=52000,
  n.burnin=2000,
  n.chains=5,
  n.thin=1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 124
##    Unobserved stochastic nodes: 124
##    Total graph size: 373
##
## Initializing model
```

```
#head(VacuumLogN.sim$BUGSoutput$sims.matrix)
length(VacuumLogN.sim$BUGSoutput$sims.matrix[,1])
```

```
## [1] 250000
```

```
mu <- VacuumLogN.sim$BUGSoutput$sims.matrix[,2]
tau <- VacuumLogN.sim$BUGSoutput$sims.matrix[,3]

gelman.diag(VacuumLogN.sim$BUGSoutput,multivariate=F)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## deviance       1.00       1.00
## mu             1.01       1.01
## tau            1.01       1.02
```

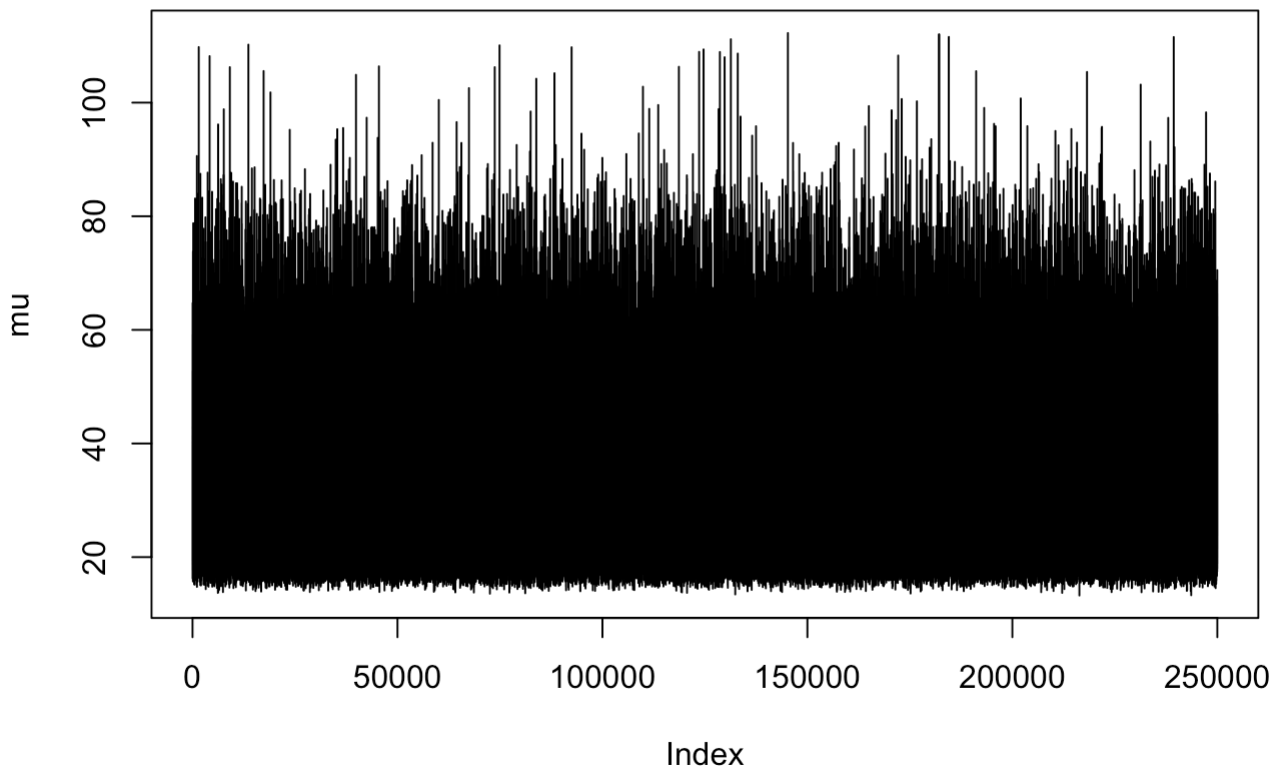
```
VacuumLogN.sim$BUGSoutput$DIC
```

```
## [1] 27.5075
```

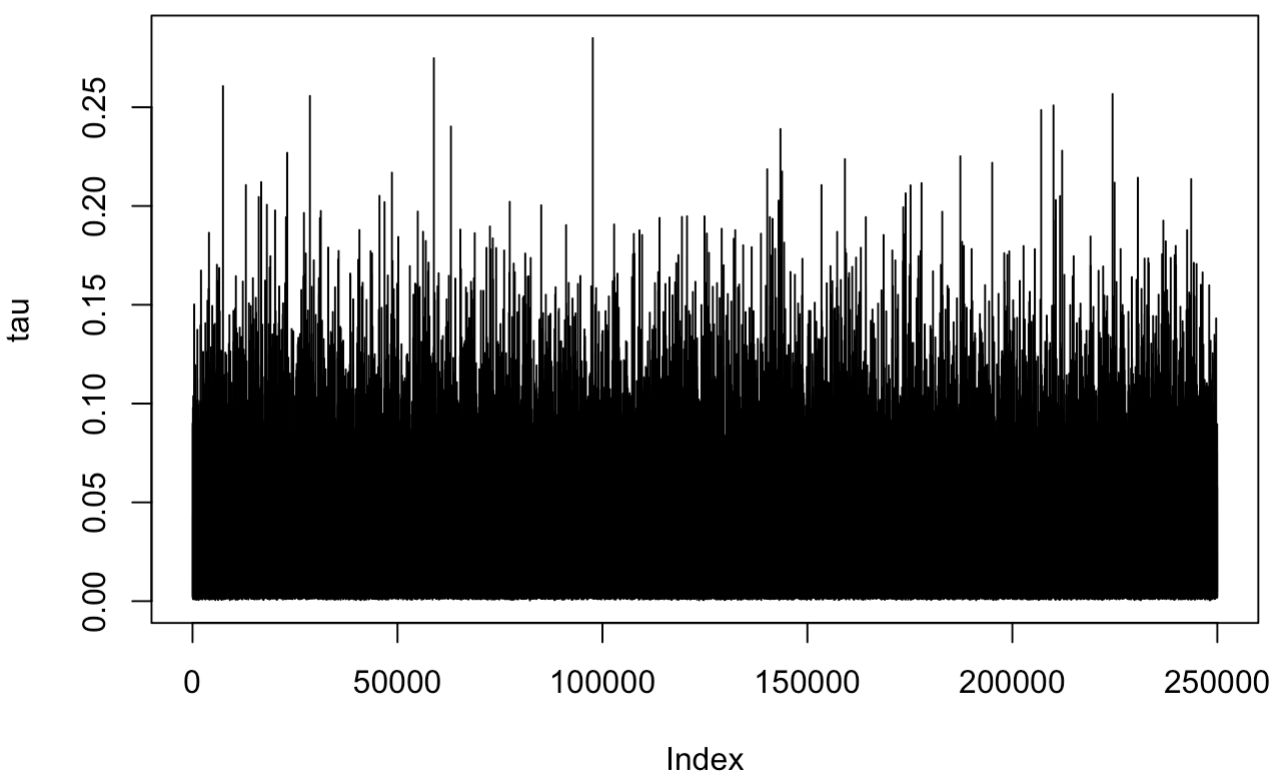
```
effectiveSize(VacuumLogN.sim)
```

```
##   deviance         mu        tau
## 11919.0127   486.1732   897.6734
```

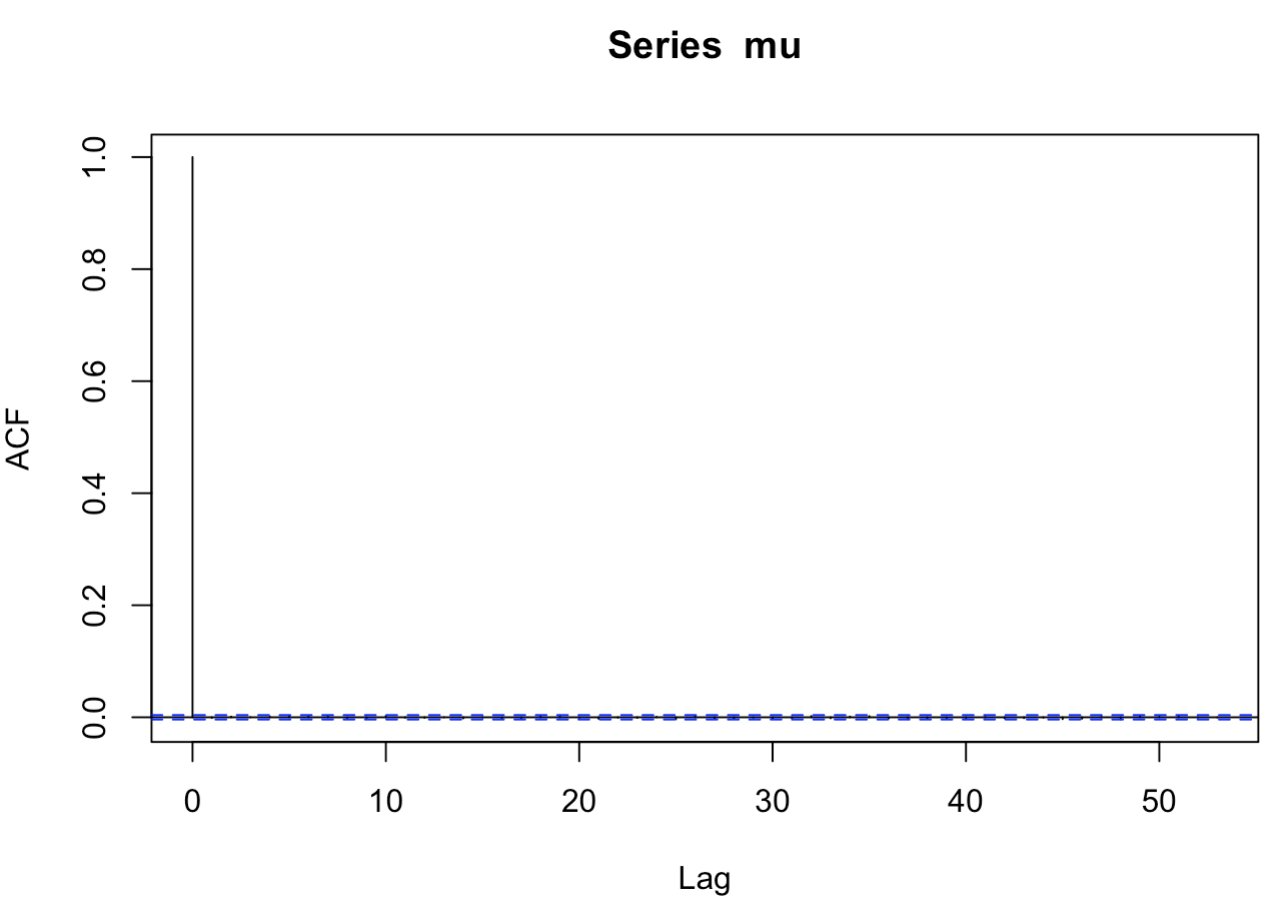
```
plot(mu, type='l')
```



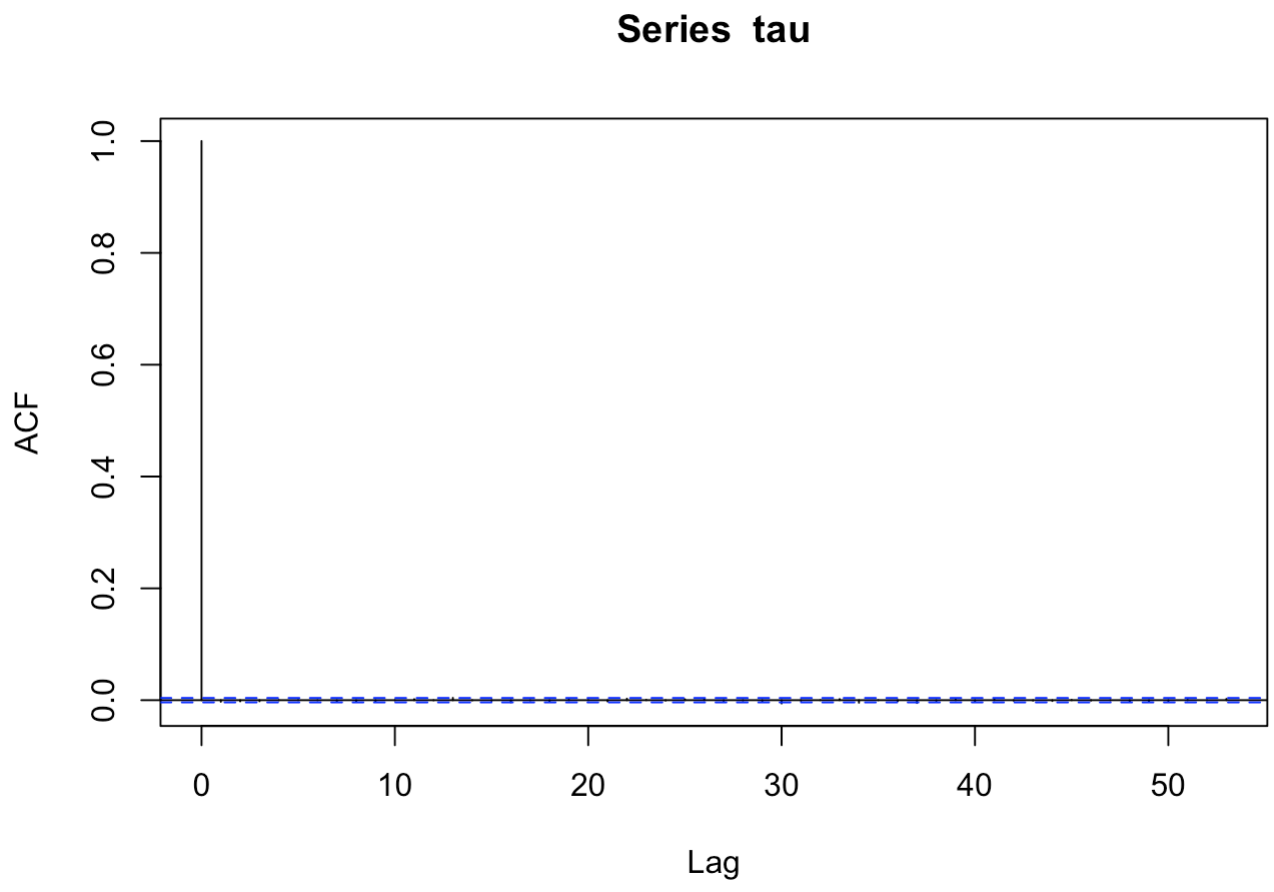

```
plot(tau, type='l')
```



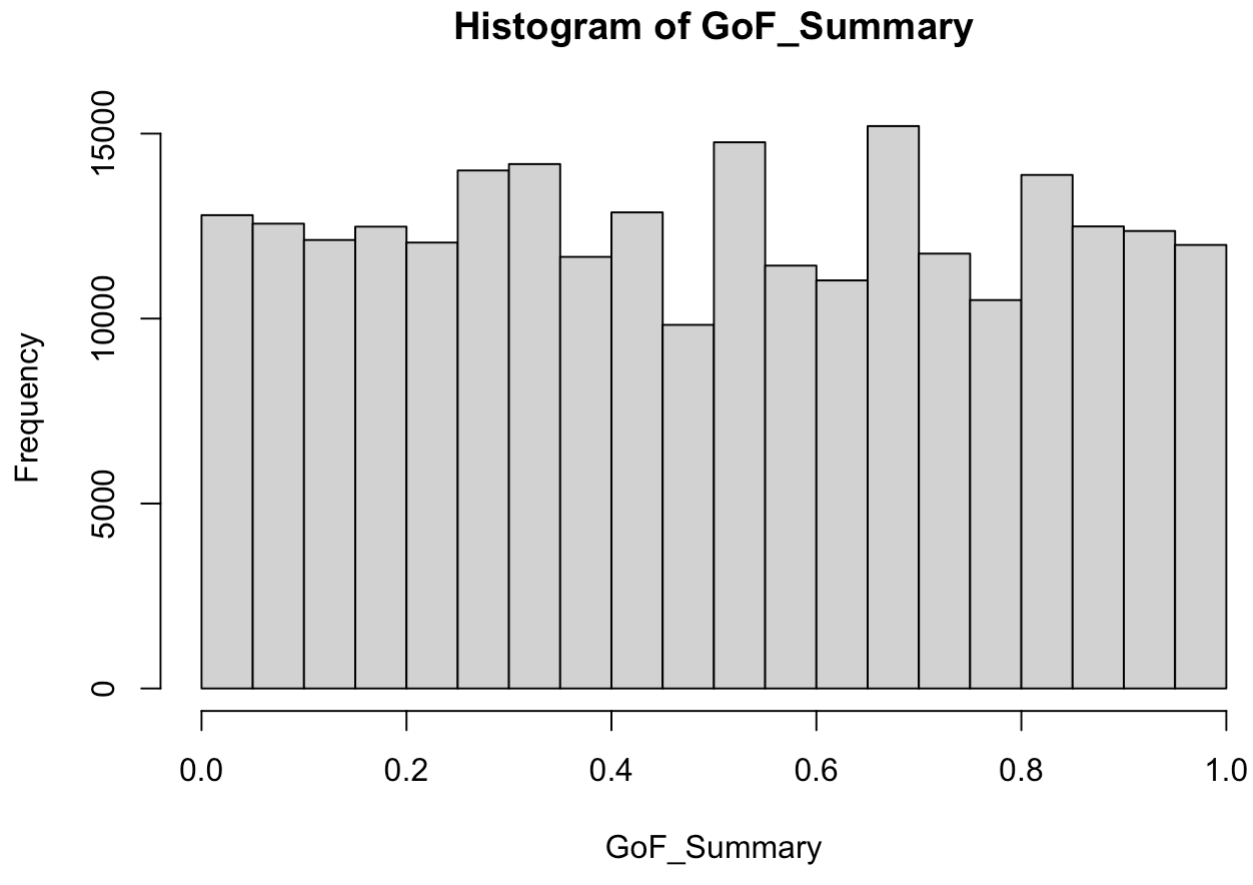

```
acf(mu)
```



```
acf(tau)
```




```
GoF <- matrix(NA,ncol=length(vac.fail),nrow=length(mu))
for (i in 1:length(mu)) {
  GoF[i,] <- plnorm(vac.fail,mu[i],1/sqrt(tau[i]))
  temp <- runif(length(vac.fail),GoF[i,],1)
  GoF[i,] <- pmax(GoF[i,],temp*I)
}

# Function requires fitted quantiles and returns a p-value
GoF_Test <- function(fitted_quantiles) {
  n <- length(fitted_quantiles)
  K <- round((n)^(0.4))
  mK <- table(cut(fitted_quantiles,(0:K)/K))
  np <- n/K
  RB <- sum(((mK-np)^2)/np)
  return(1-pchisq(RB,K-1))
}

# Calculating the p-values for each posterior model
GoF_Summary <- apply(GoF,1,GoF_Test)

# Histogram of posterior model p-values
hist(GoF_Summary,xlim=c(0,1))
```




```
# Percent of posterior models with p-value less than 0.05
mean(GoF_Summary < 0.05)
```

```
## [1] 0.051184
```

```
VacuumGG <- "model {
  for(i in 1:123){
    I[i] ~ dinterval(t[i],c[i])
    t[i] ~ dgamma(lambda, b)
  }
  lambda ~ dexp(0.001)
  b ~ dexp(0.001)
}
"

VacuumGG.sim <- jags(
  data=c('t','c','I'),
  parameters.to.save=c('lambda','b'),
  model.file=textConnection(VacuumGG),
  n.iter=22000,
  n.burnin=2000,
  n.chains=5,
  n.thin=1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 124
##    Unobserved stochastic nodes: 124
##    Total graph size: 372
##
## Initializing model
```

```
#head(VacuumGG.sim$BUGSoutput$sims.matrix)
length(VacuumGG.sim$BUGSoutput$sims.matrix[,1])
```

```
## [1] 100000
```

```
lambda <- VacuumGG.sim$BUGSoutput$sims.matrix[,3]
b <- VacuumGG.sim$BUGSoutput$sims.matrix[,1]

gelman.diag(VacuumGG.sim$BUGSoutput,multivariate=F)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## b              1.02       1.05
## deviance       1.01       1.03
## lambda         1.02       1.06
```

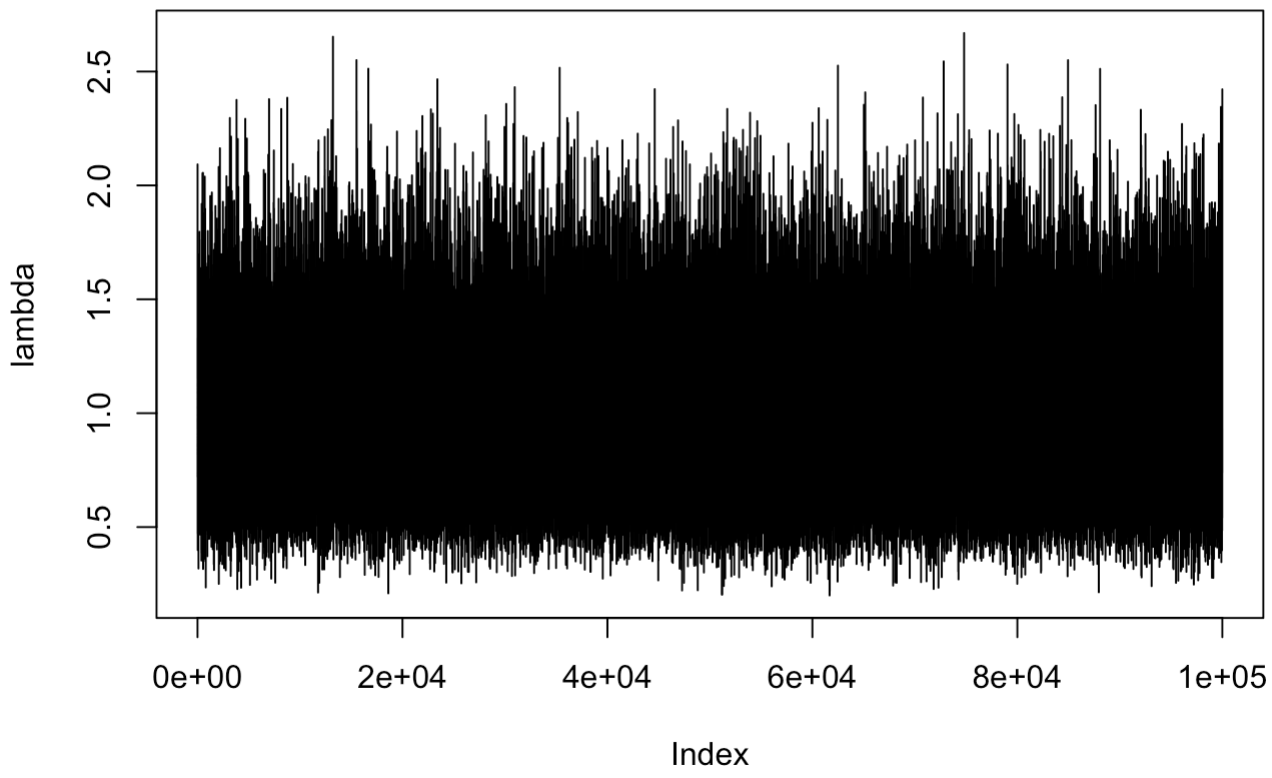
```
VacuumGG.sim$BUGSoutput$DIC
```

```
## [1] 34.98468
```

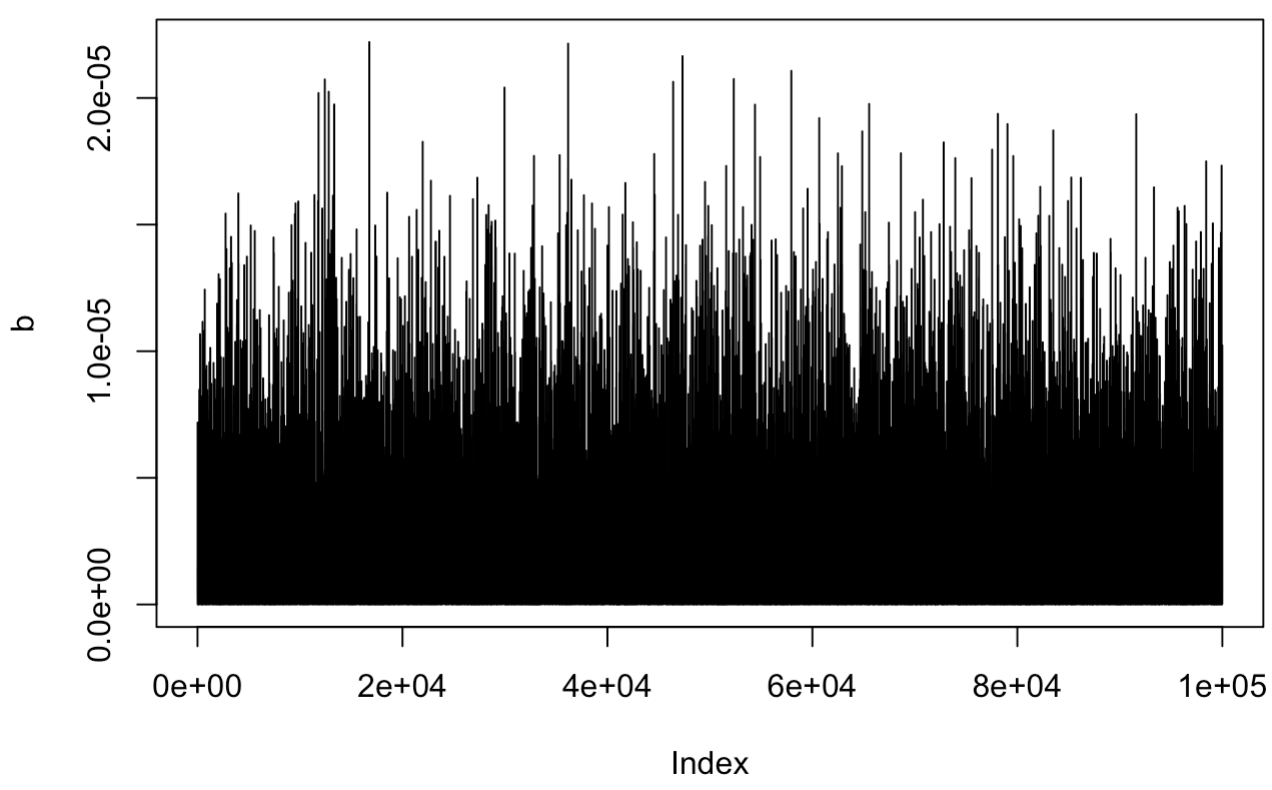
```
effectiveSize(VacuumGG.sim)
```

```
##         b  deviance    lambda
##  360.3952 1390.3139  392.4900
```

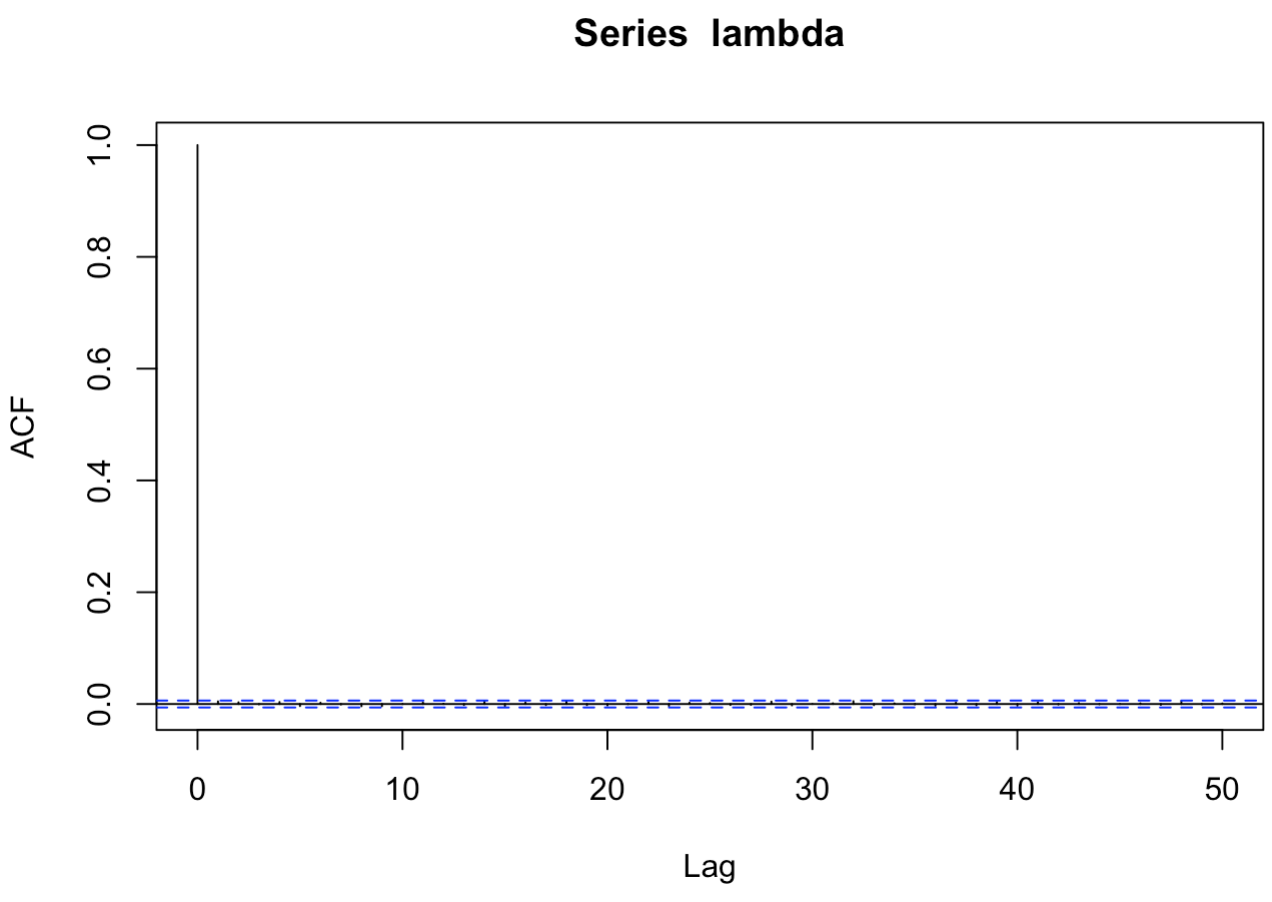
```
plot(lambda, type='l')
```



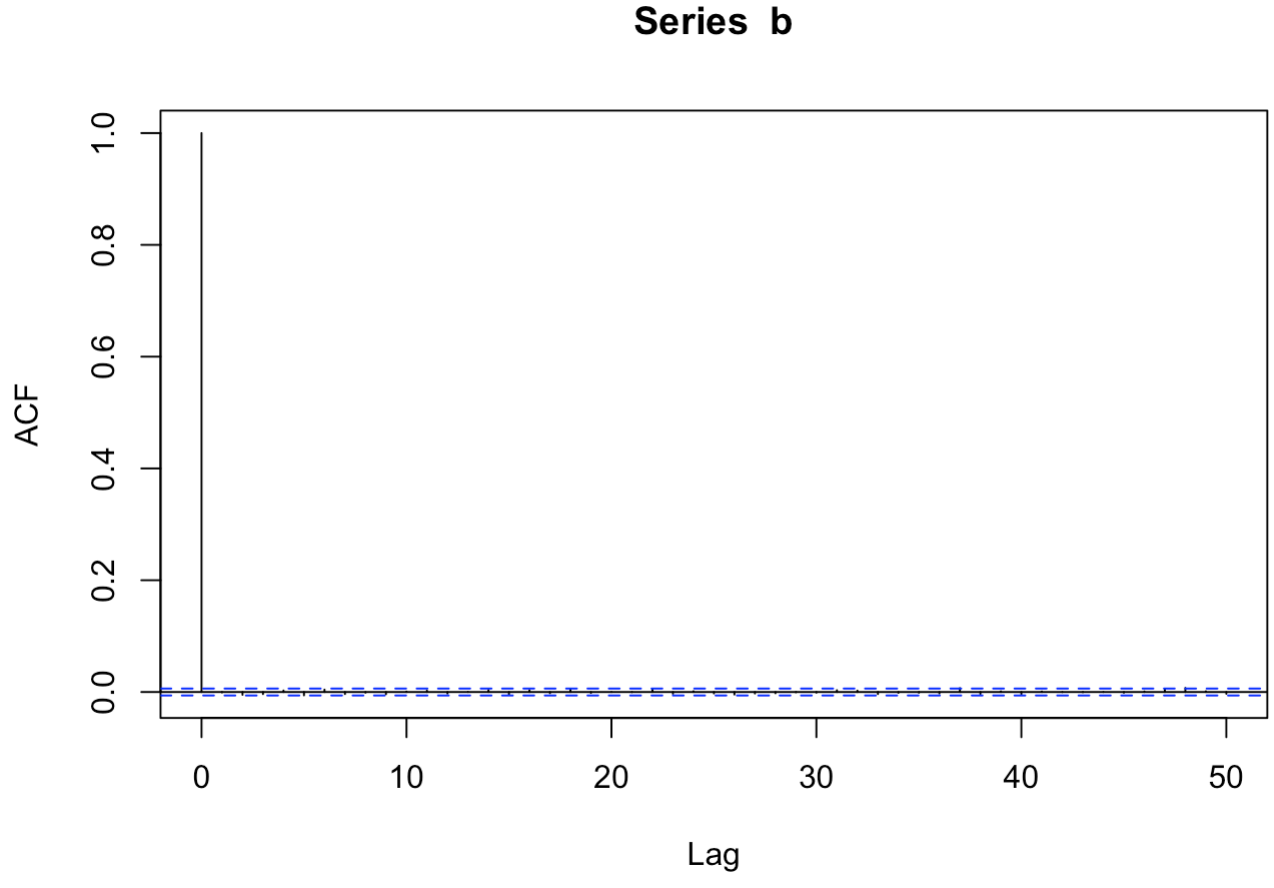

```
plot(b, type='l')
```



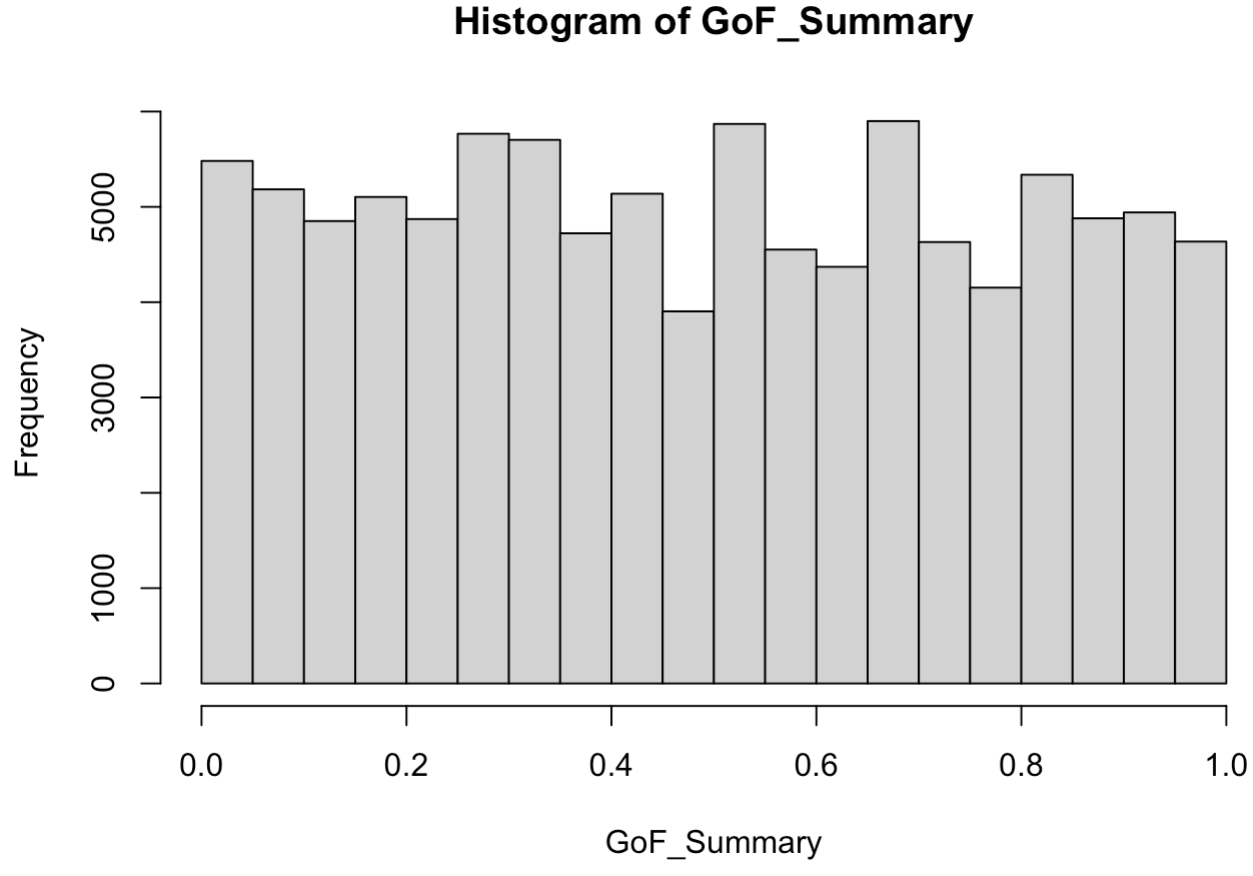

```
acf(lambda)
```



```
acf(b)
```




```
GoF <- matrix(NA,ncol=length(vac.fail),nrow=length(lambda))
for (i in 1:length(lambda)) {
  GoF[i,] <- pgamma(vac.fail, lambda[i], b[i])
  temp <- runif(length(vac.fail),GoF[i,],1)
  GoF[i,] <- pmax(GoF[i,],temp*I)
}

# Function requires fitted quantiles and returns a p-value
GoF_Test <- function(fitted_quantiles) {
  n <- length(fitted_quantiles)
  K <- round((n)^(0.4))
  mK <- table(cut(fitted_quantiles,(0:K)/K))
  np <- n/K
  RB <- sum(((mK-np)^2)/np)
  return(1-pchisq(RB,K-1))
}

# Calculating the p-values for each posterior model
GoF_Summary <- apply(GoF,1,GoF_Test)

# Histogram of posterior model p-values
hist(GoF_Summary,xlim=c(0,1))
```




```
# Percent of posterior models with p-value less than 0.05
mean(GoF_Summary < 0.05)
```

```
## [1] 0.05482
```

Log-Normal model has the lowest DIC, making it the best model. The Goodness-of-Fit is similar for the exponential, log-normal and gamma models.