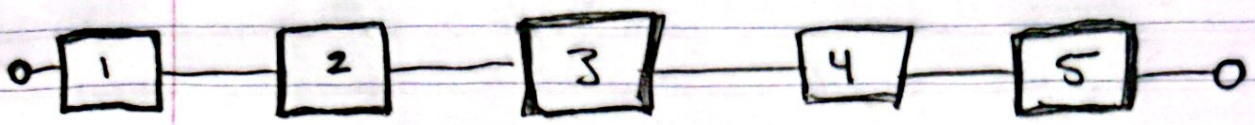
STAT 466 HW #12 04/16/24

#1 5.1



Making cereal for breakfast:

1. get out a bowl & spoon
2. pour cereal in the bowl
3. milk in the bowl
4. eat cereal
5. wash dishes

* cereal comes first, don't fight me on that

$$F_{system} = 1 - \left[(1-F_1(t))(1-F_2(t))(1-F_3(t))(1-F_4(t))(1-F_5(t))\right]$$

#2 5.8 $\quad h(t) = \left(\frac{f(t)}{R(t)}\right)^n$

$f(t)$ w/ $\theta = 0 \Rightarrow \lambda\beta(x)^{\beta-1} e^{-\lambda(x)^\beta} \quad \lambda, \beta > 0$

$$R(t) = 1 - \left[\int_0^t \lambda\beta\, x^{\beta-1} e^{-\lambda x^\beta}\, dx\right]$$

$$\lambda\beta \int_0^t x^{\beta-1} e^{-\lambda x^\beta}$$

$(x^\beta)(x^{-1})\left(e^{-\lambda x^\beta}\right)$

$u = x^\beta$

$du = \beta x^{\beta-1}\, dx$

$dx = \frac{1}{\beta x^{\beta-1}}\, du$

$x=0 \quad u = x = t^{\beta}$

$$= \int \left(e^{-\lambda u}\right)\left(x^{\beta-1}\right)\left(\frac{1}{\beta x^{\beta-1}}\right) du = \int_0^{t^{\beta}} \frac{e^{-\lambda u}}{\beta} du$$

$$\beta\lambda \frac{1}{\beta} \int_0^{t^{\beta}} e^{-\lambda u} du \Rightarrow \lambda \left[\frac{e^{-\lambda u}}{-\lambda}\right]\Bigg|_0^{t^{\beta}}$$

$$= -\left[e^{\lambda u}\right]\Big|_0^{t^{\beta}} = -\left[e^{-\lambda t^{\beta}} - 1\right]$$

$$h(t) = \left[\frac{f(t)}{R(t)}\right]^n$$

$$\text{CDF} = -e^{-\lambda t^{\beta}} + 1$$

$$\text{Reliability} = 1 + e^{-\lambda t^{\beta}} - 1$$

$$= e^{-\lambda t^{\beta}}$$

$$= \frac{\prod_{i=1}^{n} \lambda\beta x^{\beta-1} e^{-\lambda x^{\beta}}}{\prod_{i=1}^{n} e^{-\lambda x^{\beta}}}$$

$$h(t) = \prod_{i=1}^{n} \left[\lambda\beta x^{\beta-1}\right] = \lambda^n \beta^n x^{\sum_{i=1}^{n} \beta_i - 1}$$

STAT 466 HW 12

Brody Anderson

2024-04-16

Question 1

Draw a reliability block diagram describing how to successfully perform an everyday task. Include an equation which shows the system CDF comprised of the component CDFs.

Question 2

Calculate the hazard function for a series system with n components when each component lifetime has a Weibull distribution. (use Parameterization 1 from Appendix B, with $\theta = 0$)

Question 3

Fit a log-linear process to the data in Table 6.2. How does the fit compare with that of a PLP in terms of a Bayesian $\chi 2$ goodness-of-fit test? Also compare the log-linear process and PLP fits using DIC discussed in Sect. 4.6.

```
super.comp <- read.csv('table62.txt', header=TRUE, sep = '', strip.white = TRUE)
head(super.comp)
```

```
##   Month Cumulative.Day Failures Cumulative.Fail
## 1   Jul             31        5               5
## 2   Aug             62        4               9
## 3   Sep             92        6              15
## 4   Nov            123        1              16
## 5   Dec            153        2              18
## 6   Jan            184        1              19
```

```
cum.fail <- super.comp$Cumulative.Fail
cum.day <- super.comp$Cumulative.Day
failures <- super.comp$Failures
n <- length(failures)
z <- rep(0,n)

BMSmpLLM.HW <- "model {
  for(i in 1:(n-1)){
    cum.fail[i] ~ dpois(lam[i])
    lam[i] <- -(gam0+gam1*cum.day[i]) + C
  }
  cum.fail[n] ~ dpois(lam[n])
  lam[n] = exp(gam0)*(exp(gam1*cum.day[n])-1)/gam1 + C
  gam0 ~ dnorm(0,1/10)
  gam1 ~ dnorm(0,1/10)
  C = 1000
}
"

jags.inits <- list(list(gam0=1,gam1=1),
                   list(gam0=0,gam1=-1),
                   list(gam0=-1,gam1=-1))

BMSmpLLM.HW.sim <- jags(
  data=c('cum.day','cum.fail','n'),
  parameters.to.save=c('gam0','gam1'),
  model.file=textConnection(BMSmpLLM.HW),
  inits=jags.inits,
  n.iter=32000,
  n.burnin=2000,
  n.chains=3,
  n.thin=1
)
```

```
## module glm loaded
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 15
##    Unobserved stochastic nodes: 2
##    Total graph size: 101
##
## Initializing model
```

```
head(BMSmpLLM.HW.sim$BUGSoutput$sims.matrix)
```

```
##      deviance      gam0          gam1
## [1,] 26883.78 -3.293488  0.0002808562
## [2,] 26883.47 -3.513959 -0.0007114971
## [3,] 26883.82 -4.015364  0.0015455789
## [4,] 26886.85 -2.558481 -0.0040811540
## [5,] 26883.62 -3.057508 -0.0018198002
## [6,] 26883.20 -3.656092  0.0010440093
```

```
length(BMSmpLLM.HW.sim$BUGSoutput$sims.matrix[,1]) #90,000 samples obtained
```

```
## [1] 90000
```

```
gam0 <- BMSmpLLM.HW.sim$BUGSoutput$sims.matrix[,2]
gam1 <- BMSmpLLM.HW.sim$BUGSoutput$sims.matrix[,3]

gelman.diag(BMSmpLLM.HW.sim$BUGSoutput)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## deviance          1          1
## gam0              1          1
## gam1              1          1
##
## Multivariate psrf
##
## 1
```

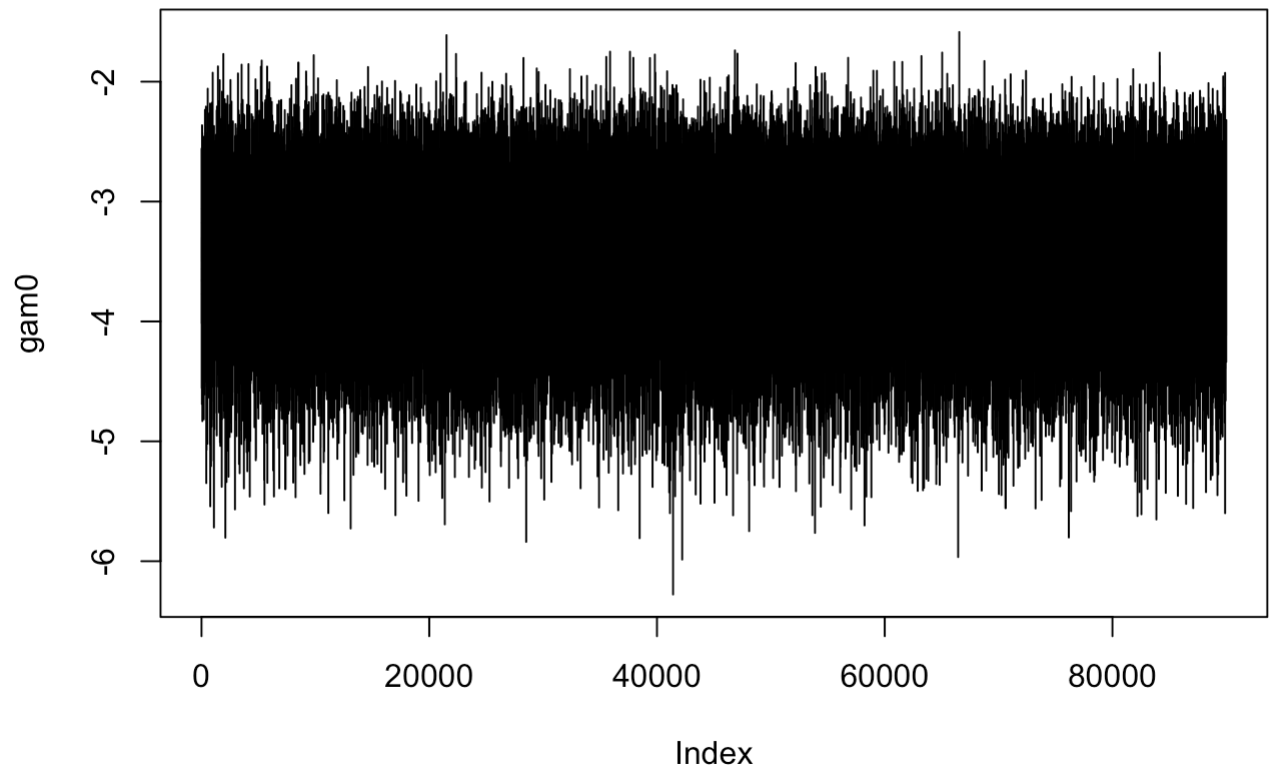
```
BMSmpLLM.HW.sim$BUGSoutput$DIC
```

```
## [1] 26886.85
```

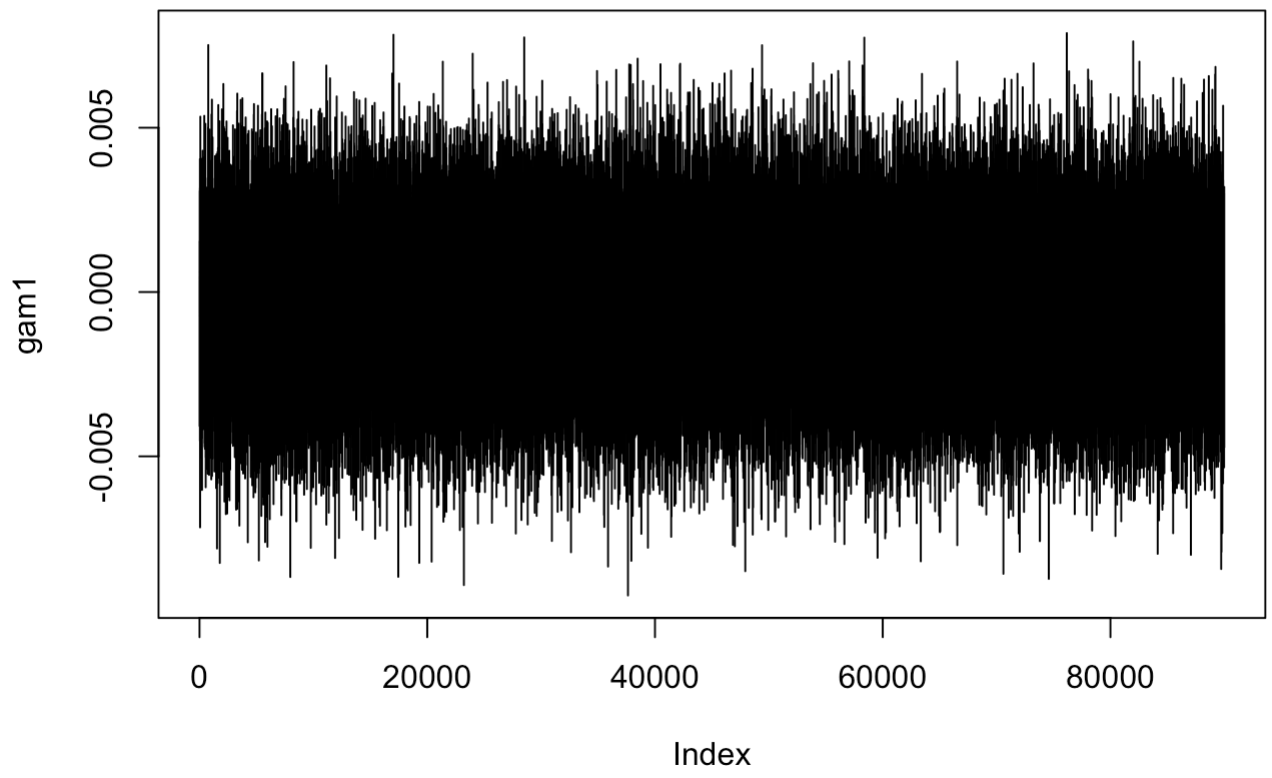
```
effectiveSize(BMSmpLLM.HW.sim)
```

```
##  deviance      gam0      gam1
## 19649.474  8221.810  8298.632
```

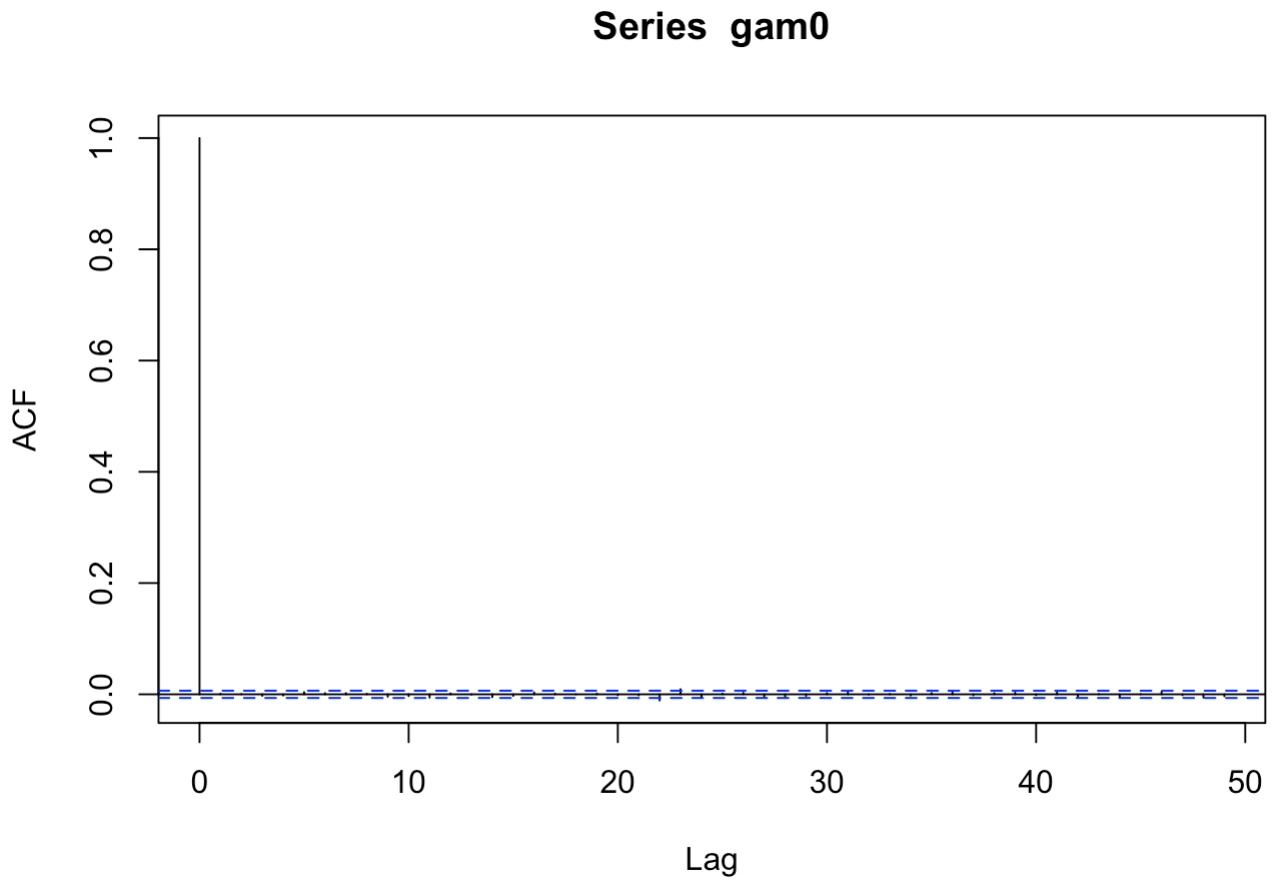
```
plot(gam0, type='l')
```



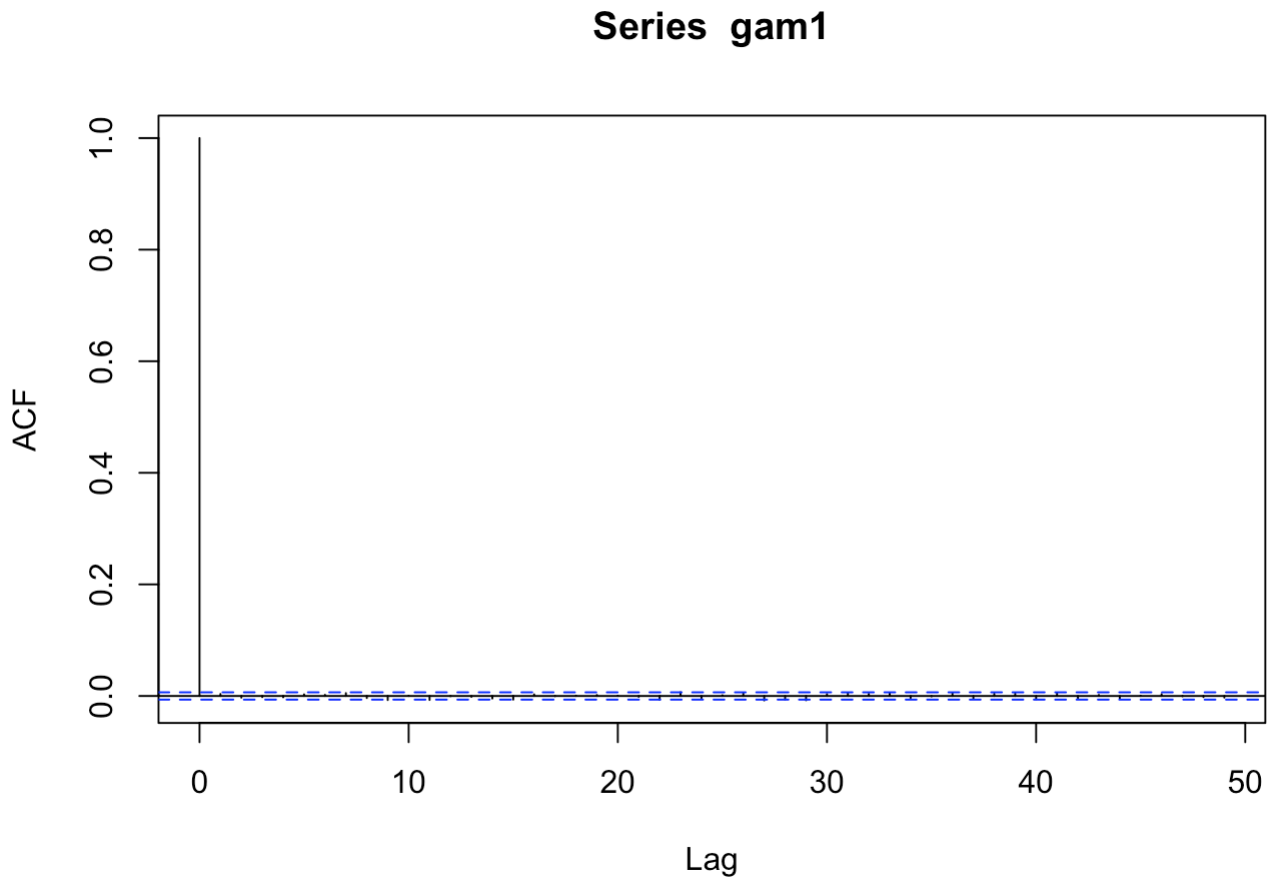

```
plot(gam1, type='l')
```




```
acf(gam0)
```




```
acf(gam1)
```



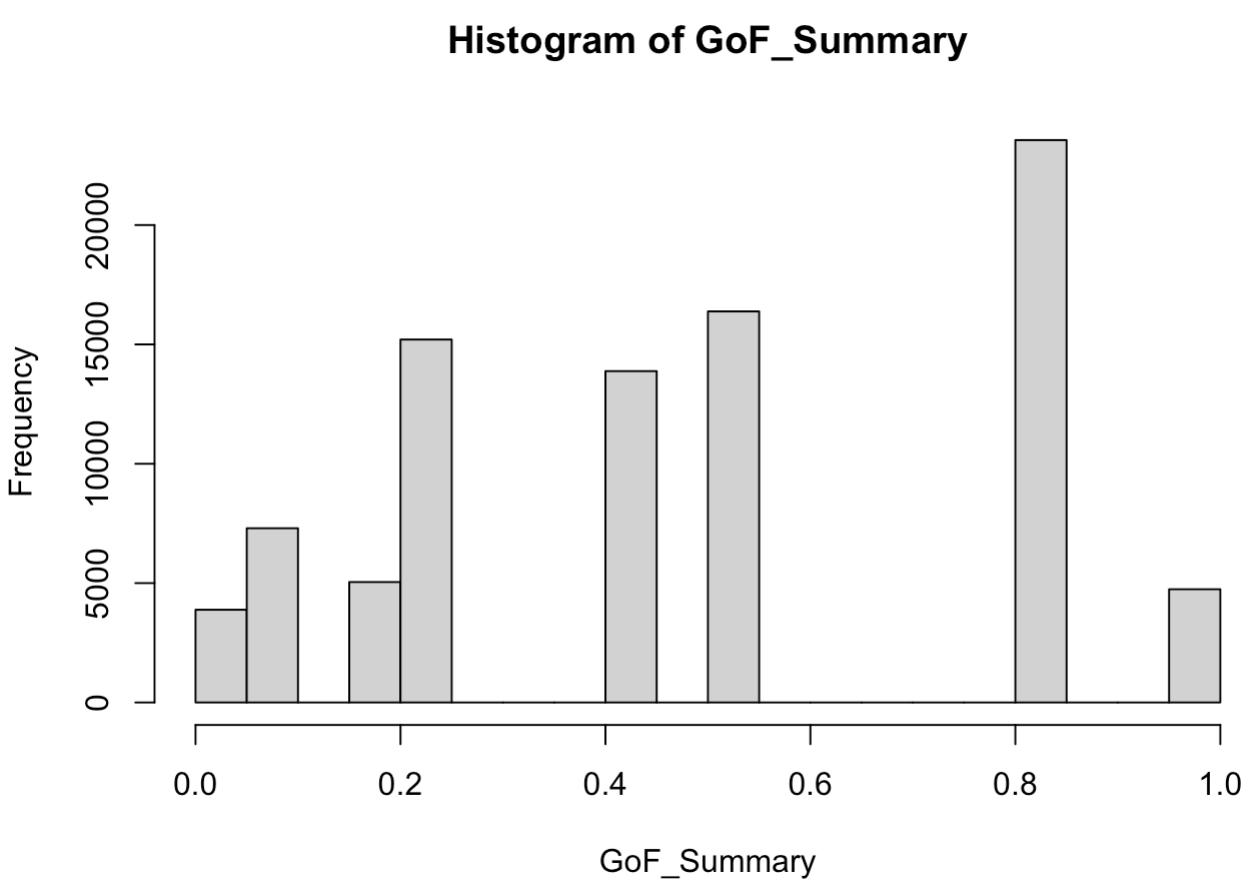
```
mcmcChain <- BMSmpLLM.HW.sim$BUGSoutput$sims.matrix[,3]
C <- 1000

GoF <- matrix(NA,ncol=length(cum.fail),nrow=length(mcmcChain))
for (i in 1:length(mcmcChain)) {
  sim.fail <- numeric(length(cum.fail))
    for (j in 1:length(cum.fail)) { #exp(gam0)*(exp(gam1*cum.day[n])-1)/gam1 + C
      sim.fail[j] <- exp(gam0[i])*(exp(gam1[i]*cum.day[j])-1)/gam1[i] + C
    }
  GoF[i,] <- ppois(cum.fail,sim.fail)
  temp <- runif(length(cum.fail),GoF[i,],1)
  GoF[i,] <- pmax(GoF[i,],temp)
}


# 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))
}

GoF_Summary <- apply(GoF,1,GoF_Test)

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




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

```
## [1] 0.04315556
```

```
# Power Law Process
cum.fail <- super.comp$Cumulative.Fail
cum.day <- super.comp$Cumulative.Day
failures <- super.comp$Failures
n <- length(cum.day)
z <- rep(0,n)

BMSmpPLP <- "model {
  for (i in 1:(n-1)) {
    z[i] ~ dpois(lam[i])
    lam[i] = -log(L[i]) + C
    L[i] = phi/eta * (cum.fail[i]/eta)^(phi-1)
  }
  z[n] ~ dpois(lam[n])
  lam[n] = L[n]
  L[n] = (cum.fail[n]/eta)^(phi)
  eta ~ dexp(1)
  phi ~ dexp(1)
  C = 100
}
"

BMSmpPLP.sim <- jags(
  data = c('cum.fail', 'z', 'n'),
  parameters.to.save = c('eta', 'phi'),
  model.file = textConnection(BMSmpPLP),
  n.iter = 32000,
  n.burnin = 2000,
  n.chains = 3,
  n.thin = 1
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 15
##    Unobserved stochastic nodes: 2
##    Total graph size: 123
##
## Initializing model
```

```
head(BMSmpPLP.sim$BUGSoutput$sims.matrix)
```

```
##      deviance      eta      phi
## [1,] 2853.809 2.876219 1.208531
## [2,] 2856.138 1.406667 0.843632
## [3,] 2853.466 3.548596 1.062315
## [4,] 2851.409 4.741009 1.474377
## [5,] 2851.229 4.001995 1.258556
## [6,] 2852.588 2.927208 1.127695
```

```
length(BMSmpPLP.sim$BUGSoutput$sims.matrix[,1]) #90,000 samples obtained
```

```
## [1] 90000
```

```
eta <- BMSmpPLP.sim$BUGSoutput$sims.matrix[,2]
phi <- BMSmpPLP.sim$BUGSoutput$sims.matrix[,3]

gelman.diag(BMSmpPLP.sim$BUGSoutput)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## deviance          1          1
## eta               1          1
## phi               1          1
##
## Multivariate psrf
##
## 1
```

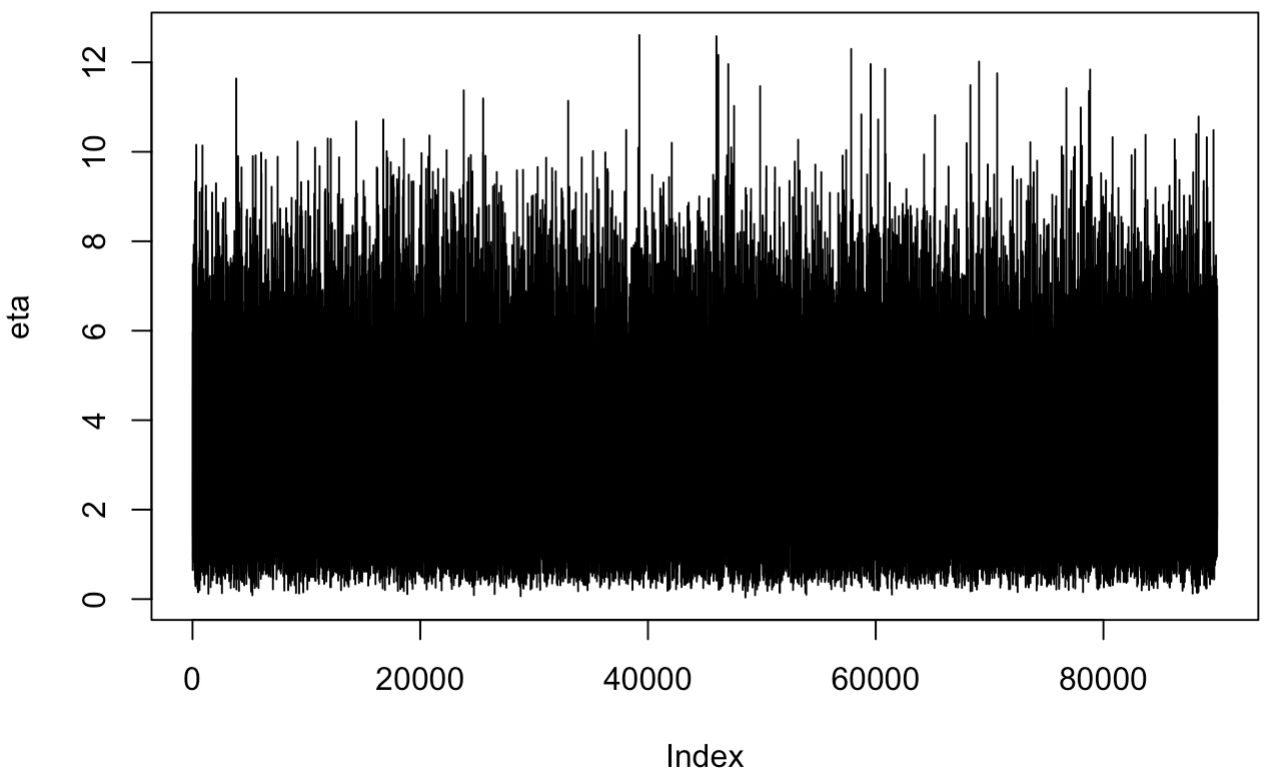
```
BMSmpPLP.sim$BUGSoutput$DIC
```

```
## [1] 2858.168
```

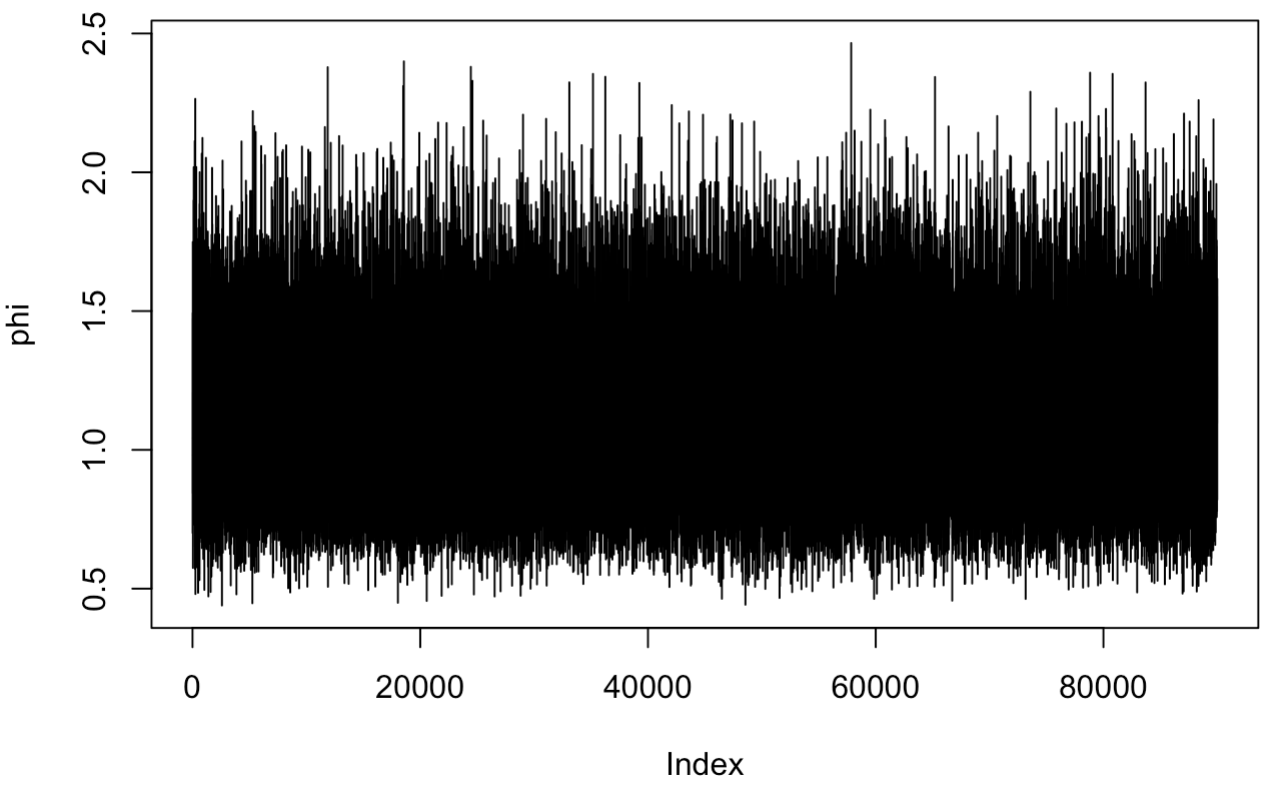
```
effectiveSize(BMSmpPLP.sim)
```

```
## deviance      eta      phi
## 6078.055 4429.091 4514.328
```

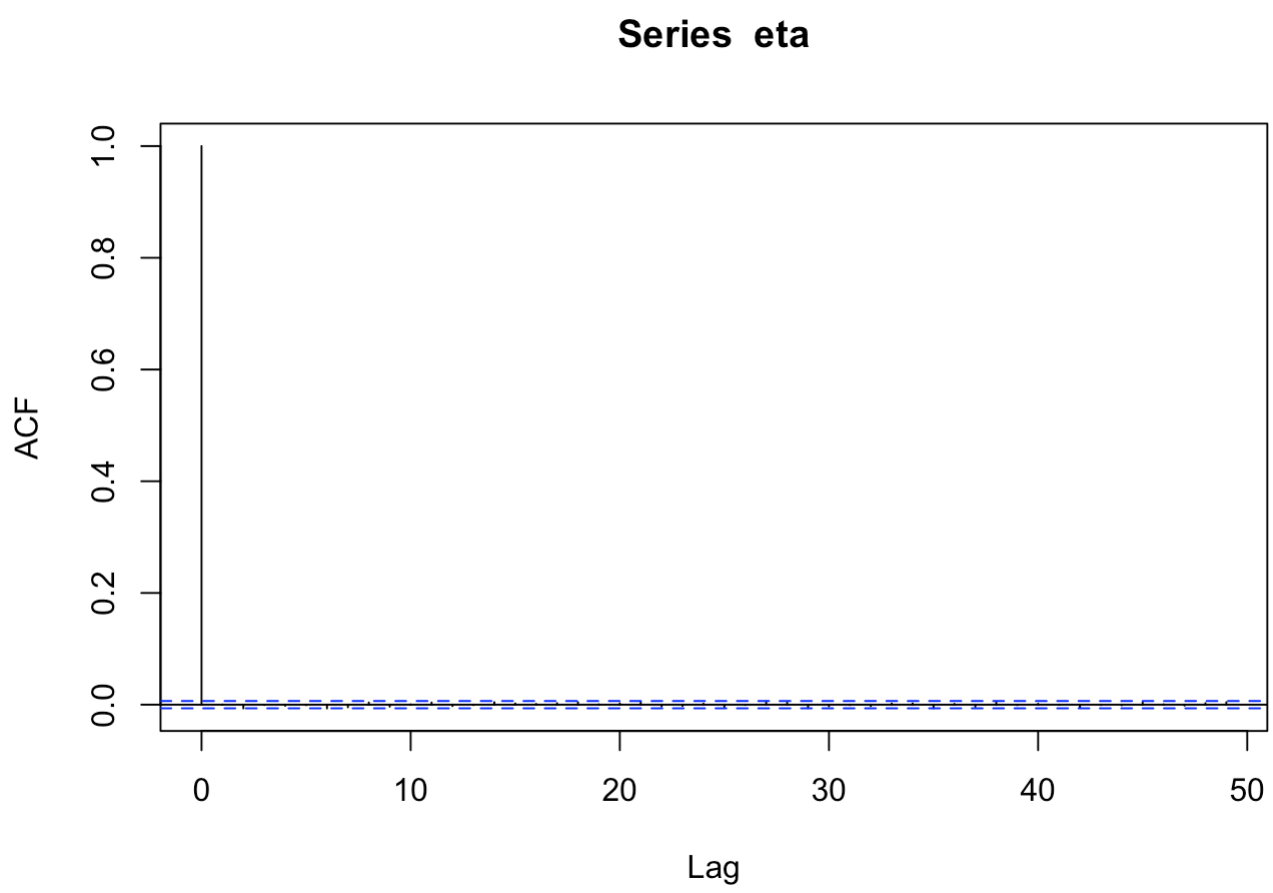
```
plot(eta, type='l')
```



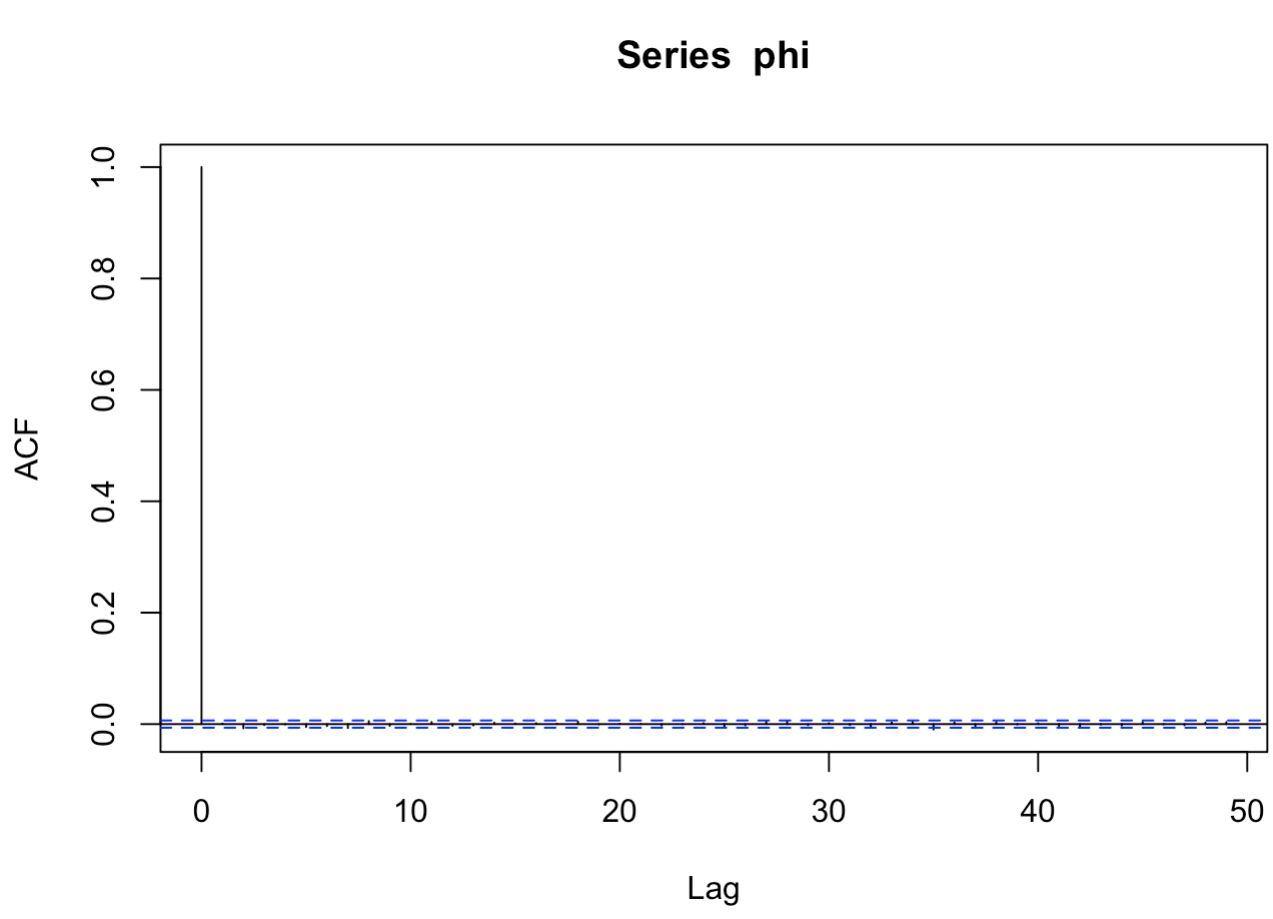

```
plot(phi, type='l')
```




```
acf(eta)
```


Series eta
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
Lag

```
acf(phi)
```


Series phi
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
Lag

```
mcmcChainPLP <- BMSmpPLP.sim$BUGSoutput$sims.matrix[,3]

GoF <- matrix(NA,ncol=length(cum.fail),nrow=length(mcmcChainPLP))
for (i in 1:length(mcmcChainPLP)) {
  sim.fail <- numeric(length(cum.fail))
  for (j in 1:length(cum.fail)) { #phi/eta * (cum.fail[i]/eta)^(phi-1)
    sim.fail[j] <- (phi[i]/eta[i])*(cum.fail[j]/eta[i])^(phi[i]-1)
  }
  GoF[i,] <- ppois(cum.fail,sim.fail)
  temp <- runif(length(cum.fail),GoF[i,],1)
  GoF[i,] <- pmax(GoF[i,],temp)
}

# 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))
}

GoF_Summary <- apply(GoF,1,GoF_Test)

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

#mean(GoF_Summary < 0.05)
# couldn't get this to run properly
```

DIC for the PLP is lower, but the log-linear process GoF is really good. The model fits the data well.