Applied Bayesian Analysis : NCSU ST 540

Midterm 3 TEST WF

Bruce Campbell

```
 library(fitdistrplus)
  library(gamlss)
setwd("c:/e/brucebcampbell-git/bayesian-learning-with-R/E3")
load("heatwaves.RData")
n.chains =2;
nSamples = 10000
load("HWD2.RData")
k =5
# Fit JAGS GLM Models for Poisson
library(rjags)
library(coda)
model_code = '
model
{
    ## Likelihood
    for(i in 1:N){
      Y[i] ~ dpois(lambda[i])
      log(lambda[i]) <- mu[i]
      mu[i] <- intercept + beta*t[i]
    }

  ## Priors
  beta ~ dnorm(mu.beta,tau.beta)
  intercept ~ dnorm(mu.intercept,tau.intercept)

  ## Posterior Predictive Checks
  for(i in 1:N){
    Y2[i]    ~ dpois(lambda[i])
  }
  D[1] <- mean(Y2[])
  D[2] <- sd(Y2[])
}
'
print(paste("JAGS GLM Models for Poisson ", k,sep = " "))
```

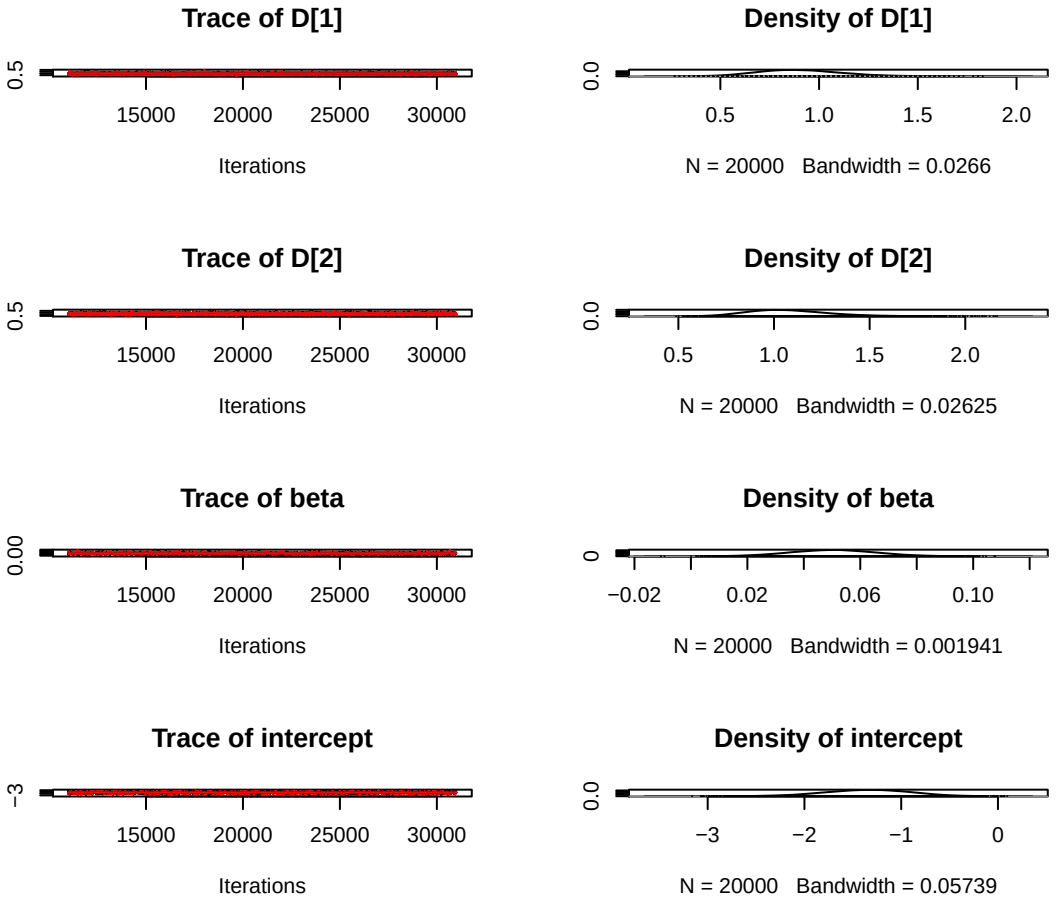
```
## [1] "JAGS GLM Models for Poisson  5"
```

```
  # Set up the data
  model_data = list(N = 41, t=seq(1:41),Y=X.num[,k],mu.beta=0,tau.beta=.0001,mu.intercept=0,tau
  # Choose the parameters to watch
  model_parameters =  c("beta", "intercept","D")
```

```
  model <- jags.model(textConnection(model_code),data = model_data,n.chains = n.chains)#Compi
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 41
##    Unobserved stochastic nodes: 43
##    Total graph size: 256
##
## Initializing model
```

```
  update(model, nSamples, progress.bar="none"); # Burnin
  out.coda  <- coda.samples(model, variable.names=model_parameters,n.iter=2*nSamples)
  plot(out.coda)
```




```
  #assess the posterior's stationarity, by looking at the Heidelberg-Welch convergence diagnos
  heidel.diag(out.coda)
```

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

```
##           Stationarity start     p-value
##           test         iteration
## D[1]      passed       1         0.519
## D[2]      passed       1         0.971
## beta      passed       1         0.917
## intercept passed       1         0.859
##
##           Halfwidth Mean    Halfwidth
##           test
## D[1]      passed     0.9043 0.00310
## D[2]      passed     1.0777 0.00379
## beta      passed     0.0501 0.00101
## intercept passed    -1.3494 0.02971
##
## [[2]]
##
##           Stationarity start     p-value
##           test         iteration
## D[1]      passed       1         0.930
## D[2]      passed       1         0.496
## beta      passed       1         0.988
## intercept passed       1         0.987
##
##           Halfwidth Mean    Halfwidth
##           test
## D[1]      passed     0.9025 0.003133
## D[2]      passed     1.0772 0.004136
## beta      passed     0.0507 0.000984
## intercept passed    -1.3681 0.029200
```

```
  # check that our chain's length is satisfactory.
  raftery.diag(out.coda)
```

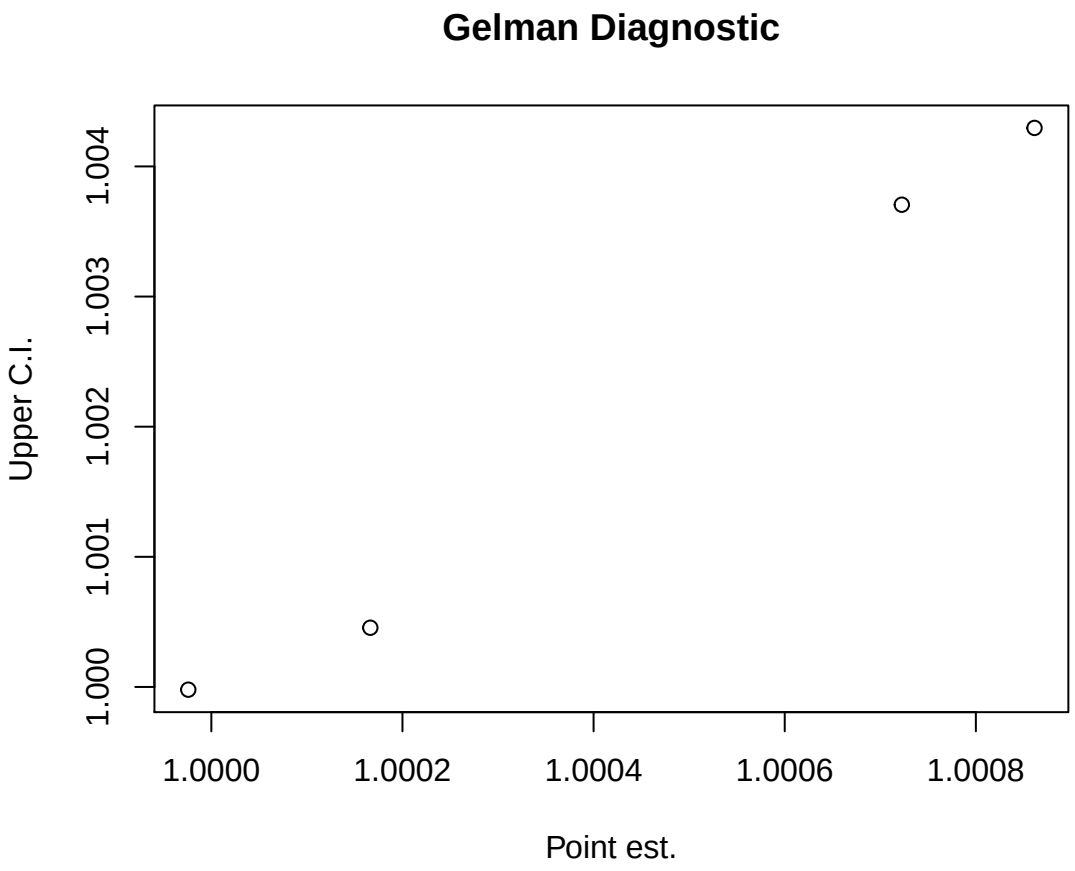
```
## [[1]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##            Burn-in  Total Lower bound  Dependence
##            (M)      (N)   (Nmin)       factor (I)
##  D[1]      2        5320  3746         1.42
##  D[2]      2        3911  3746         1.04
##  beta      18       21036 3746         5.62
##  intercept 32       33904 3746         9.05
##
##
## [[2]]
```

```
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##            Burn-in  Total Lower bound  Dependence
##            (M)      (N)   (Nmin)       factor (I)
##  D[1]      2        4778  3746         1.28
##  D[2]      2        3822  3746         1.02
##  beta      21       21603 3746         5.77
##  intercept 35       36540 3746         9.75
```

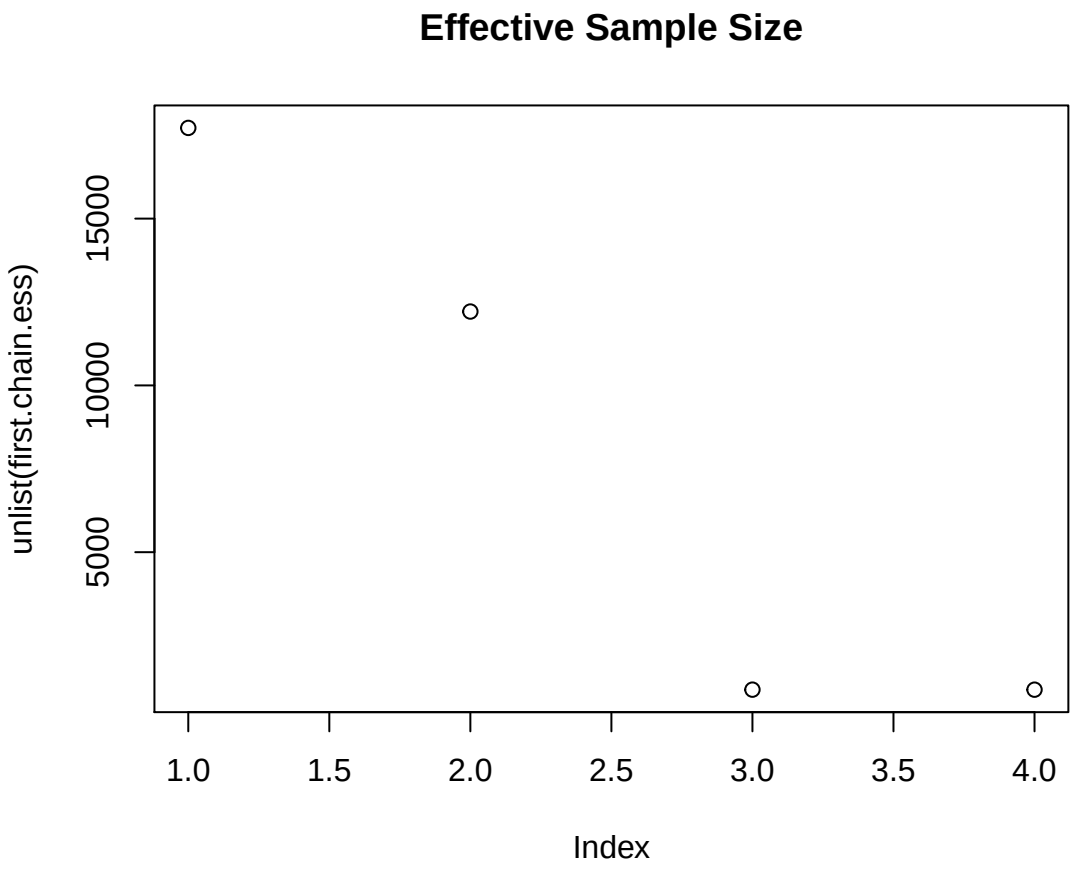
```
  geweke.diag(out.coda)
```

```
## [[1]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##      D[1]      D[2]      beta intercept
##   -0.7189    0.3035    0.6154   -0.6849
##
##
## [[2]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##      D[1]      D[2]      beta intercept
##    1.3436   -0.9058   -0.8472    0.8031
```

```
  if(n.chains > 1)
  {
    gelman.srf <-gelman.diag(out.coda)
    plot(gelman.srf$psrf,main = "Gelman Diagnostic")
  }
```




```
chains.ess <- lapply(out.coda,effectiveSize)
first.chain.ess <- chains.ess[1]
plot(unlist(first.chain.ess), main="Effective Sample Size")
```




```
# Compute the test stats for the data
D0   <- c(   mean(X.num[,k]),   sd(X.num[,k]))
Dnames <- c("mean Y", "sd Y")
# Compute the test stats for the models
chain <- out.coda[[1]]
D1   <- cbind(chain[,"D[1]"],chain[,"D[2]"])
pval1 <- rep(0,2)
names(pval1)<-Dnames
pval2 <- pval1
for(j in 1:2){
 pval1[j] <- mean(D1[,j]>D0[j])
}

pander(data.frame(p.vals = pval1) ,caption=paste("Posterior Perdictive check p-values ",k,se
```

Table 1: Posterior Perdictive check p-values 5

	p.vals
mean Y	0.4583
sd Y	0.6687

```
  chain <- out.coda[[1]]

  posterior.means <- list()

  for( i in 1:length(colnames(chain)) )
  {
    colname <- colnames(chain)[i]
    samples <- chain[,i]
    posterior.means[colname] <-mean(samples)
  }
    pander(data.frame(posterior.means))
```

D.1.	D.2.	beta	intercept
0.9043	1.078	0.05008	-1.349

```
###Frequentist


    df <- data.frame(t=seq(1:41),Y=X.num[,k])
    model.pois <- glm( Y~ t, family=poisson, df)
    sp <- summary.glm(model.pois)
    sp
```

```
##
## Call:
## glm(formula = Y ~ t, family = poisson, data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0369  -0.9135  -0.4620   0.6784   1.9942
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.32520    0.45391  -2.919  0.00351 **
## t            0.05012    0.01535   3.266  0.00109 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 47.826  on 40  degrees of freedom
## Residual deviance: 36.050  on 39  degrees of freedom
## AIC: 95.086
##
## Number of Fisher Scoring iterations: 5
```

```
####


# Fit JAGS GLM Models for Negative Binomial

model_code = '
model
{
    ## Likelihood
    for(i in 1:N){
      Y[i] ~ dnegbin(p[i],r)

      p[i] <- r/(r+lambda[i])
      log(lambda[i]) <- mu[i]
      mu[i] <- intercept + beta*t[i]
    }

  ## Priors
  beta ~ dnorm(mu.beta,tau.beta)
  intercept ~ dnorm(mu.intercept,tau.intercept)
  r ~ dunif(0,20)
  #r ~ dgamma(0.01,0.01)


  ## Posterior Predictive Checks
  for(i in 1:N){
    Y2[i]    ~ dnegbin(p[i],r)
  }
  D[1] <- mean(Y2[])
  D[2] <- sd(Y2[])

}
'
  print(paste("JAGS GLM Models for Negative Binomial ", k,sep = " "))
```

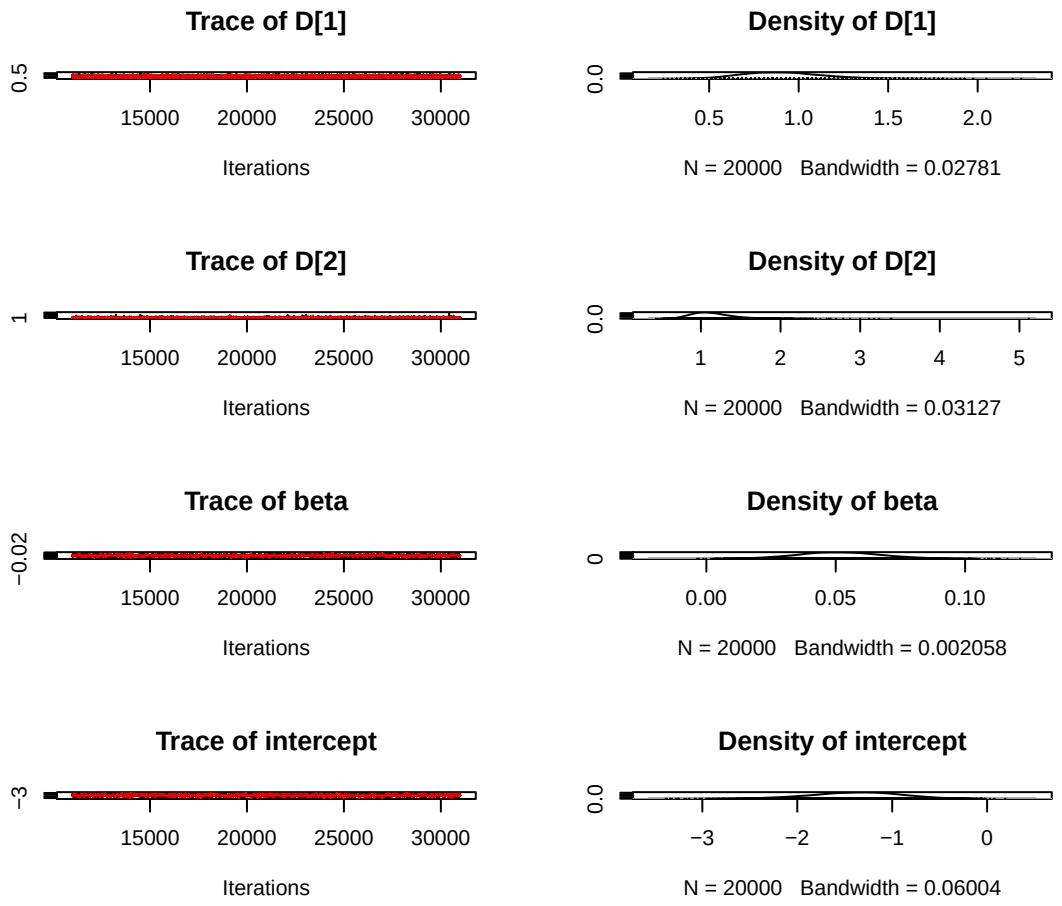
```
## [1] "JAGS GLM Models for Negative Binomial  5"
```

```
  # Set up the data
  model_data = list(N = 41, t=seq(1:41),Y=X.num[,k],mu.beta=0,tau.beta=.0001,mu.intercept=0,tau
  # Choose the parameters to watch
  model_parameters =  c("beta", "intercept","D")
   model <- jags.model(textConnection(model_code),data = model_data,n.chains = n.chains)#Compi
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 41
```

```
##    Unobserved stochastic nodes: 44
##    Total graph size: 341
##
## Initializing model
```

```
update(model, nSamples, progress.bar="none"); # Burnin
out.coda  <- coda.samples(model, variable.names=model_parameters,n.iter=2*nSamples)
plot(out.coda)
```




```
#assess the posterior's stationarity, by looking at the Heidelberg-Welch convergence diagnos
heidel.diag(out.coda)
```

```
## [[1]]
##
##           Stationarity start     p-value
##           test         iteration
## D[1]      passed       1         0.523
## D[2]      passed       1         0.370
## beta      passed       1         0.778
## intercept passed       1         0.823
##
```

```
##           Halfwidth Mean    Halfwidth
##           test
## D[1]      passed     0.9126 0.00323
## D[2]      passed     1.1390 0.00558
## beta      passed     0.0512 0.00103
## intercept passed    -1.3754 0.03005
##
## [[2]]
##
##           Stationarity start     p-value
##           test         iteration
## D[1]      passed       1         0.702
## D[2]      passed       1         0.492
## beta      passed       1         0.373
## intercept passed       1         0.431
##
##           Halfwidth Mean    Halfwidth
##           test
## D[1]      passed     0.9111 0.00322
## D[2]      passed     1.1417 0.00521
## beta      passed     0.0519 0.00104
## intercept passed    -1.3941 0.02997
```

```
  # check that our chain's length is satisfactory.
  raftery.diag(out.coda)
```

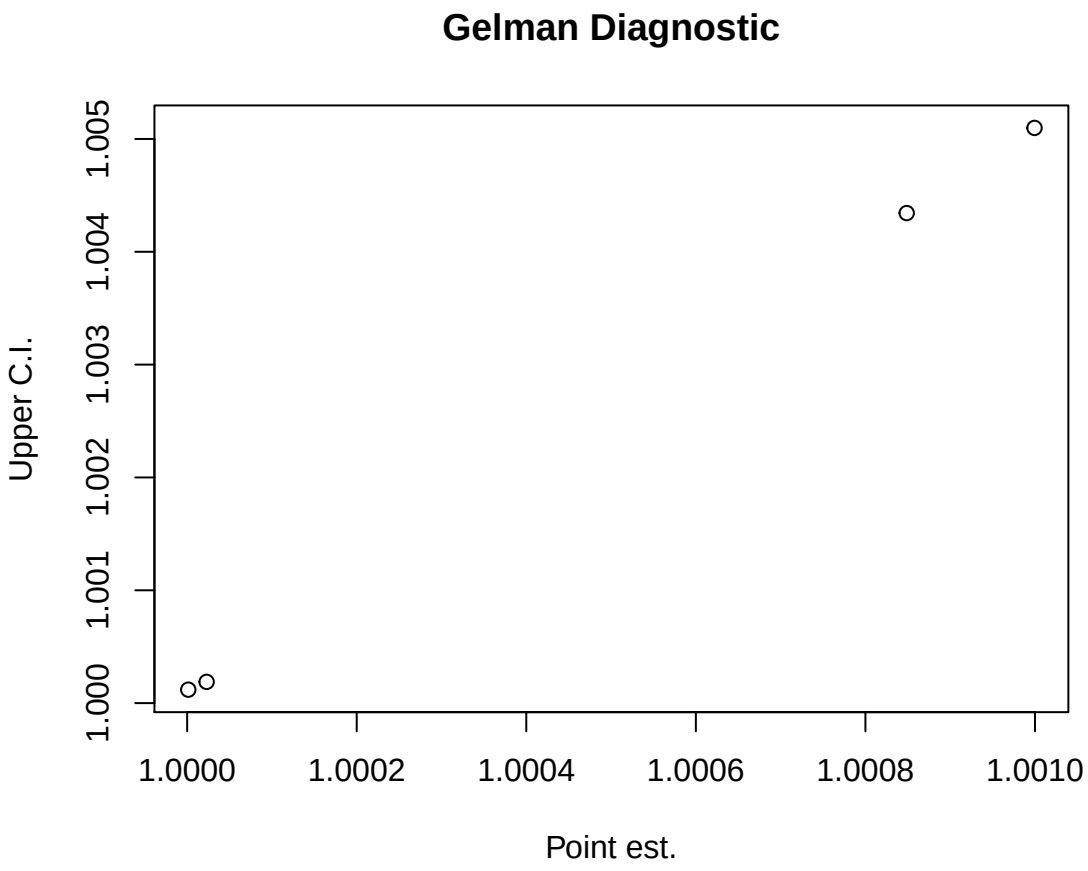
```
## [[1]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##            Burn-in  Total Lower bound  Dependence
##            (M)      (N)   (Nmin)       factor (I)
##  D[1]      2        4319  3746          1.15
##  D[2]      2        3910  3746          1.04
##  beta      21       23883 3746          6.38
##  intercept 40       39205 3746         10.50
##
##
## [[2]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##            Burn-in  Total Lower bound  Dependence
##            (M)      (N)   (Nmin)       factor (I)
```

```
##  D[1]      2        4289  3746         1.14
##  D[2]      2        3784  3746         1.01
##  beta      21       24732 3746         6.60
##  intercept 36       36582 3746         9.77
```

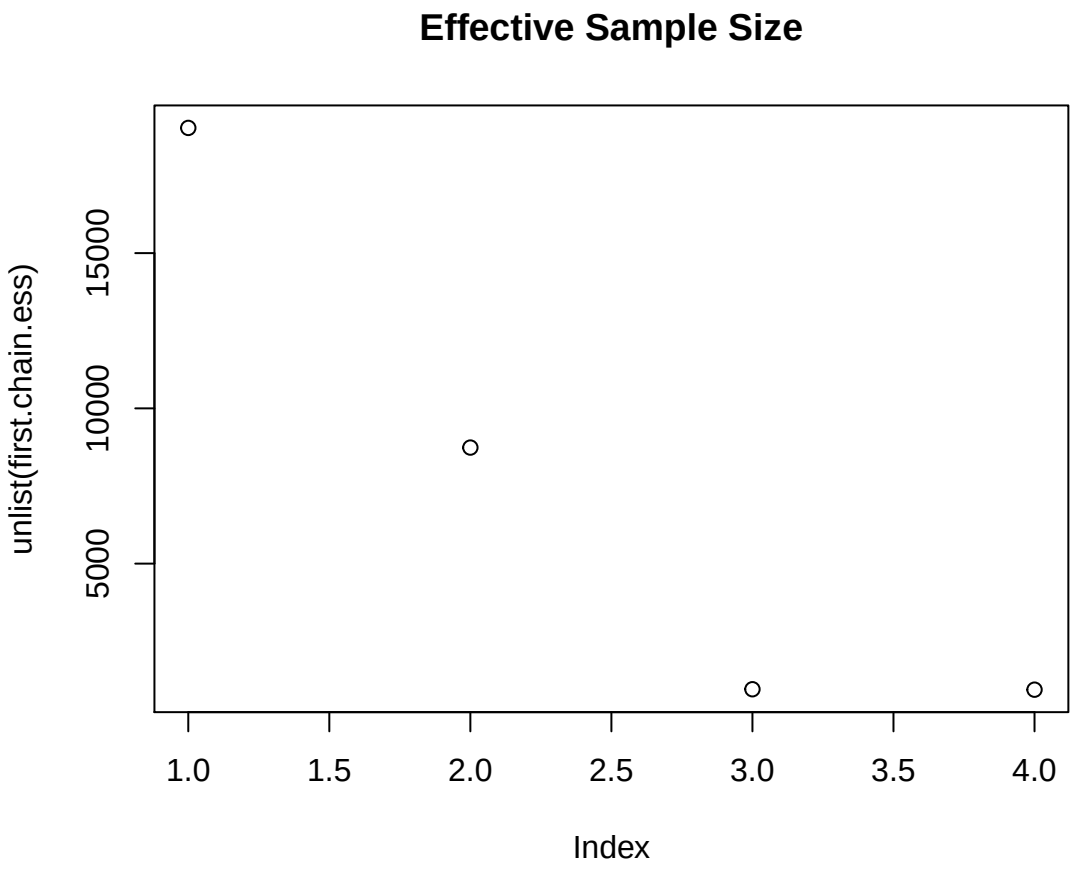
```
  geweke.diag(out.coda)
```

```
## [[1]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##      D[1]      D[2]      beta intercept
##    0.5197    0.5997    0.4482   -0.3332
##
##
## [[2]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##      D[1]      D[2]      beta intercept
##    0.2117    0.2343   -0.3837    0.2987
```

```
  if(n.chains > 1)
  {
    gelman.srf <-gelman.diag(out.coda)
    plot(gelman.srf$psrf,main = "Gelman Diagnostic")
  }
```




```
chains.ess <- lapply(out.coda,effectiveSize)
first.chain.ess <- chains.ess[1]
plot(unlist(first.chain.ess), main="Effective Sample Size")
```




```
# Compute the test stats for the data
D0   <- c(   mean(X.num[,k]),   sd(X.num[,k]))
Dnames <- c("mean Y", "sd Y")
# Compute the test stats for the models
chain <- out.coda[[1]]
D1   <- cbind(chain[,"D[1]"],chain[,"D[2]"])
pval1 <- rep(0,2)
names(pval1)<-Dnames
pval2 <- pval1
for(j in 1:2){
 pval1[j] <- mean(D1[,j]>D0[j])
}

pander(data.frame(p.vals = pval1) ,caption=paste("Posterior Perdictive check p-values ",k,se
```

Table 3: Posterior Perdictive check p-values 5

	p.vals
mean Y	0.4703
sd Y	0.7262

```
  chain <- out.coda[[1]]

  posterior.means <- list()

  for( i in 1:length(colnames(chain)) )
  {
    colname <- colnames(chain)[i]
    samples <- chain[,i]
    posterior.means[colname] <-mean(samples)
  }
    pander(data.frame(posterior.means))
```

D.1.	D.2.	beta	intercept
0.9126	1.139	0.05124	-1.375

```
### Frequentist

    model.nb <- glm.nb(Y~t,data=df)
    snb <- summary.glm(model.nb)
```