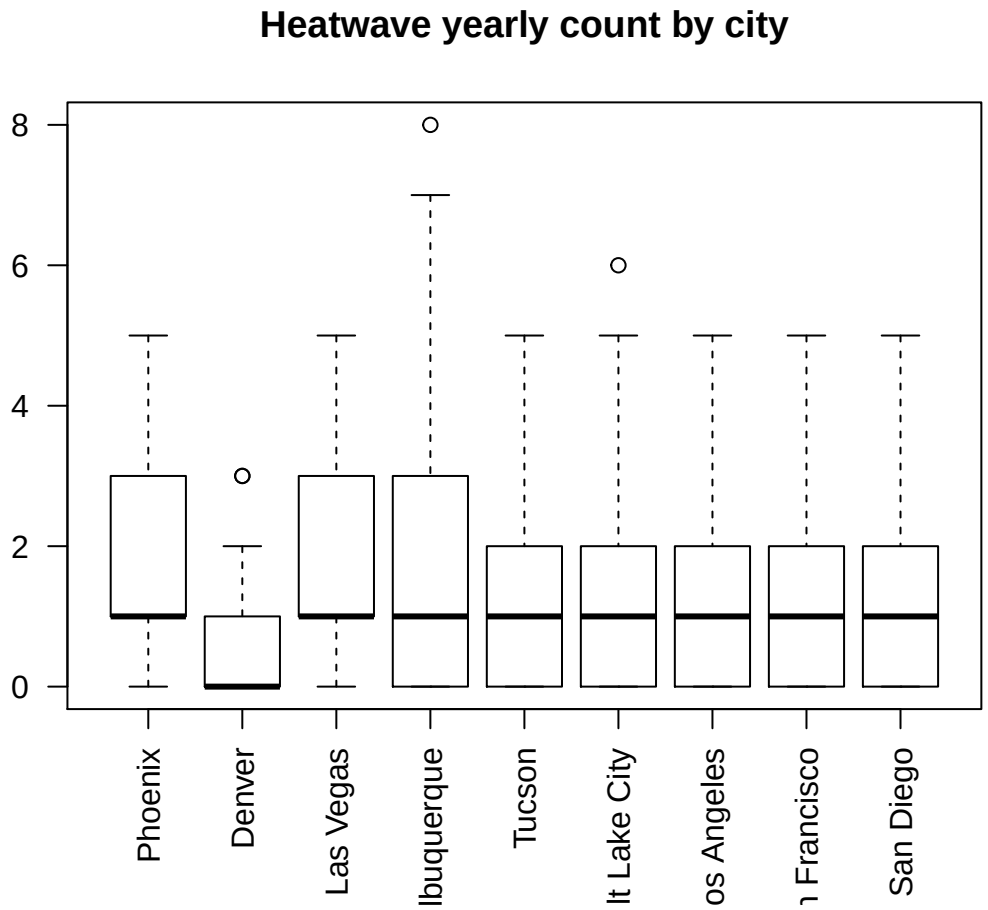
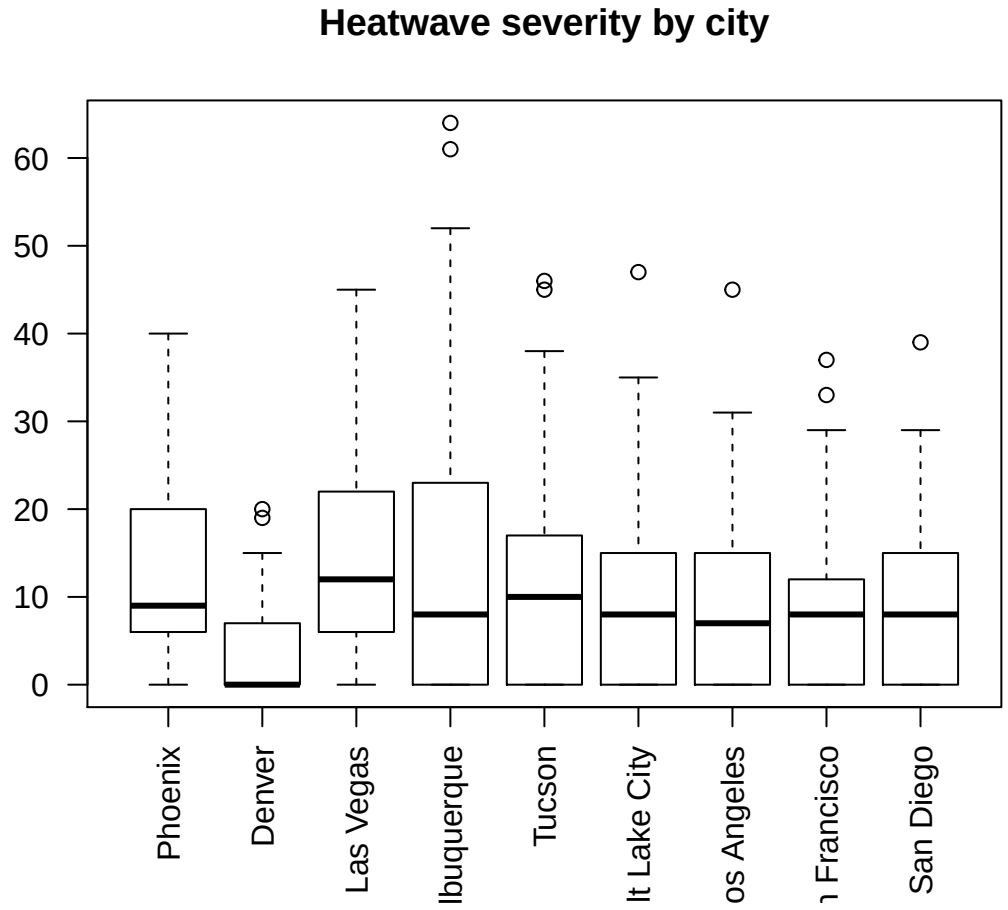
E3

```
rm(list = ls())
library(rjags)
library(coda)
library(pander)
setwd("c:/e/brucebcampbell-git/bayesian-learning-with-R/E3")
load("heatwaves.RData")
n.chains = 2
n.thin = 2
nSamples = 10000
load("HWD1.RData")

df <- data.frame(X.num)
colnames(df) <- city_names
boxplot(df, las = 2, main = "Heatwave yearly count by city")
```




```
df <- data.frame(X.sev)
colnames(df) <- city_names
boxplot(df, las = 2, main = "Heatwave severity by city")
```



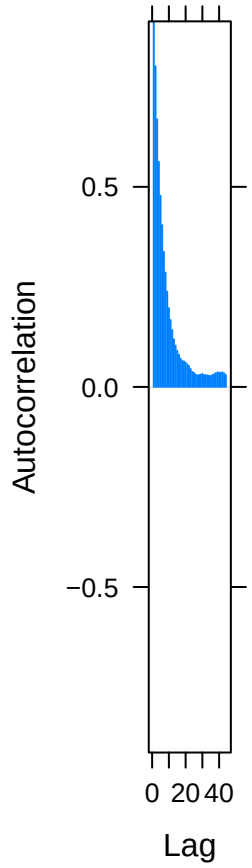
Fit JAGS Poisson Random Effects

```
######################################## Fit JAGS Poisson
model_pois = '
model
{
    ## Likelihood
    for(i in 1:N){
      for(j in 1:9){
        Y[i,j] ~ dpois(lambda[i,j])
        log(lambda[i,j]) <- mu[i,j]
        mu[i,j] <- alpha[j] + beta[j]*t[i]
      }
    }

  ## Priors
  for(i in 1:9){
    alpha[i] ~ dnorm(0,taus[i])
    taus[i] ~ dgamma(0.1,0.1)
  }
```

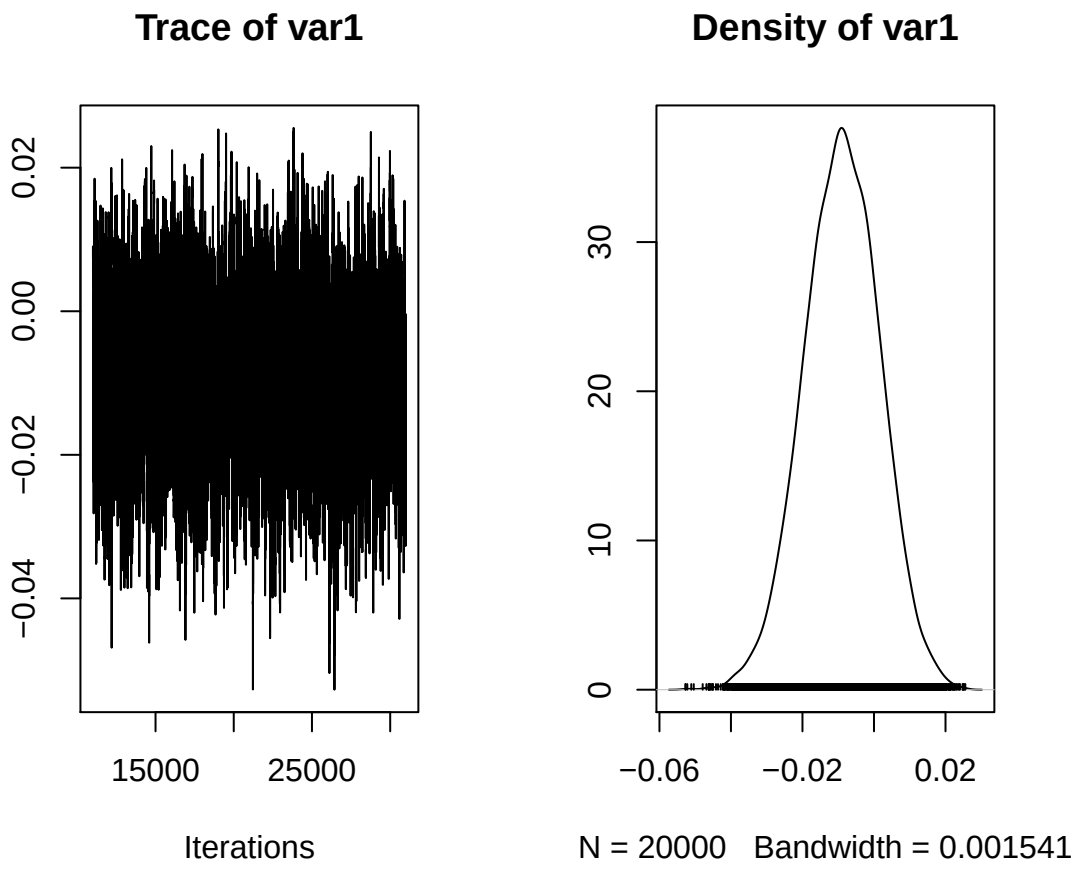
```
  # Slopes
  for(i in 1:9){
    beta[i] ~ dnorm(mu.beta,taus.beta[i])
    taus.beta[i] ~ dgamma(0.1,0.1)
  }

  ## Posterior Predictive Checks
  for(i in 1:N){
    for(j in 1:9){
        Y2[i,j] ~ dpois(lambda[i,j])
    }
  }

  for(j in 1:9){
    Dm[j] <- mean(Y2[,j])
    Dsd[j] <- sd(Y2[,j])
  }

  #Prediction
  for(i in 1:N){
    for(j in 1:9){
      Yp[i,j] ~ dpois(lambdap[i,j])
      log(lambdap[i,j]) <- mup[i,j]
      mup[i,j] <- alpha[j] + beta[j]*t[i]
    }
  }
}
'


  # Set up the data
  model_data = list(N = 41, t=seq(1:41),Y=X.num,mu.beta=0,tau.beta=.0001,mu.intercept=0,tau.intercept=.
  # Choose the parameters to watch
  model_parameters =  c("beta", "alpha","Dm", "Dsd", "Yp")
  model_pois <- jags.model(textConnection(model_pois),data = model_data,n.chains = n.chains)#Compile Mo
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 369
##    Unobserved stochastic nodes: 774
##    Total graph size: 2322
##
## Initializing model
```

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

  coda::acfplot(out.coda[[1]][,'beta[1]'],100)
```




```
plot(out.coda[[1]][,'beta[1]'])
```




```
slopes.hpd <- matrix(nrow = 9,ncol=2)
for(k in 1:9){
  coef.name <- paste('beta[', k,']', sep='')
  inv <- HPDinterval(out.coda[[1]][,coef.name],.95)
  slopes.hpd[k,]<-inv
}
colnames(slopes.hpd) <- c("lower","upper")
rownames(slopes.hpd) <-city_names
pander(data.frame(slopes.hpd), caption = "0.95 HPD Intervals for slopes")
```

Table 1: 0.95 HPD Intervals for slopes

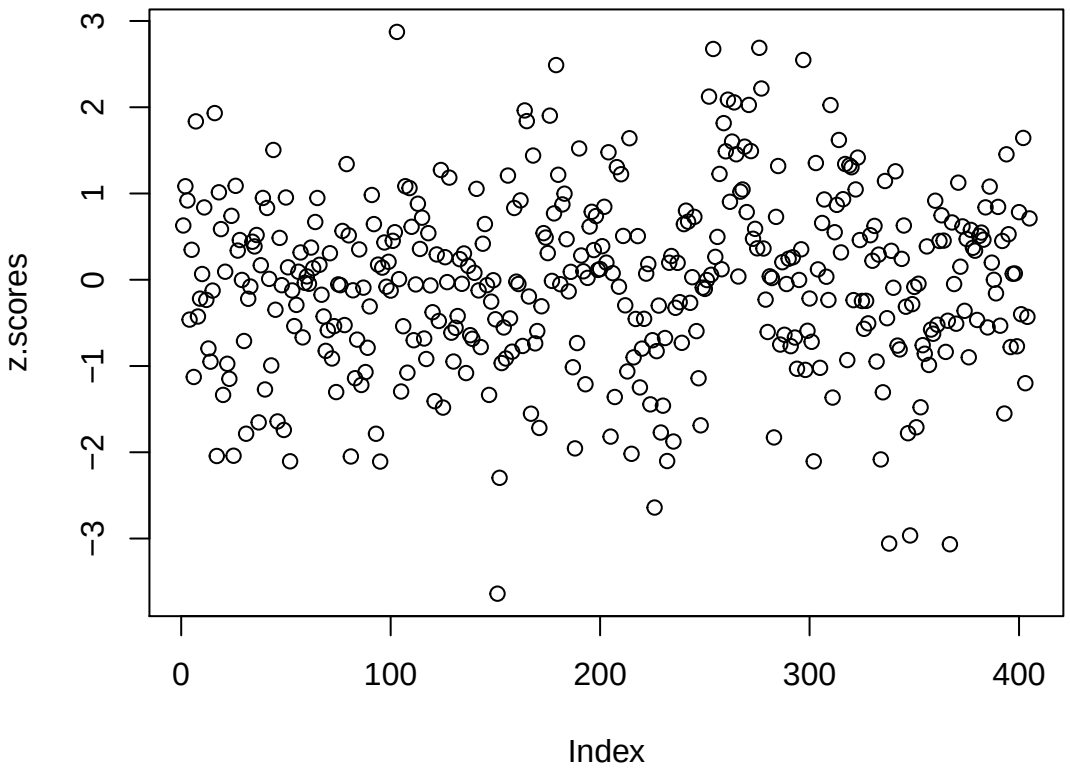
	lower	upper
Phoenix	-0.02989	0.01112
Denver	0.009508	0.079
Las Vegas	-0.008101	0.02929
Albuquerque	-0.0472	-0.007077
Tucson	0.01894	0.06065
Salt Lake City	-0.008352	0.03231
Los Angeles	-0.03171	0.01212
San Francisco	-0.01676	0.02797
San Diego	-0.003845	0.03635

```
so <-summary(out.coda)

#assess the posteriors stationarity, by looking at the Heidelberg-Welch convergence diagnostic:
hd <- heidel.diag(out.coda)
hdd <- hd[[1]]
hd.pass <- hdd[,2]
hd.fail <-sum(is.na(hd.pass))
pander(data.frame(fail.count = hd.fail), caption ="Fail count for Heidelberger and Welch diagnotic")
```

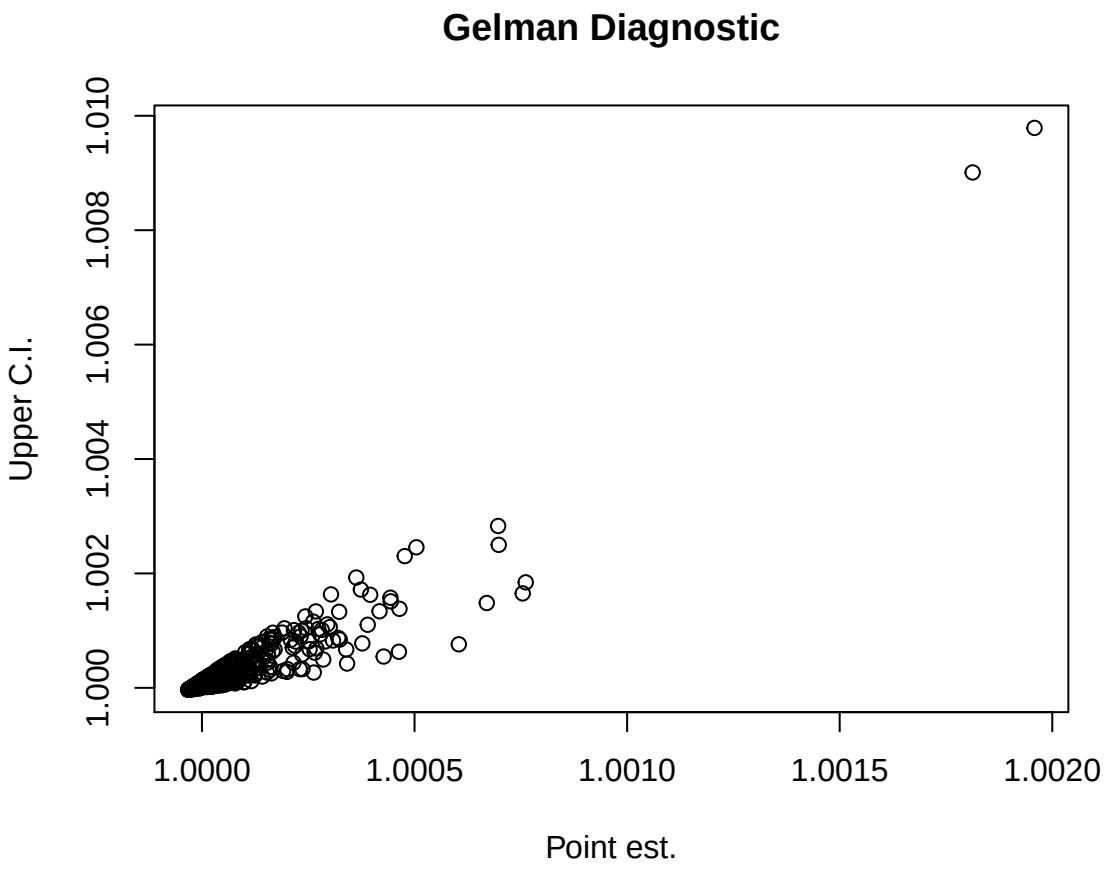
Table 2: Fail count for Heidelberger and Welch diagnotic

fail.count
4

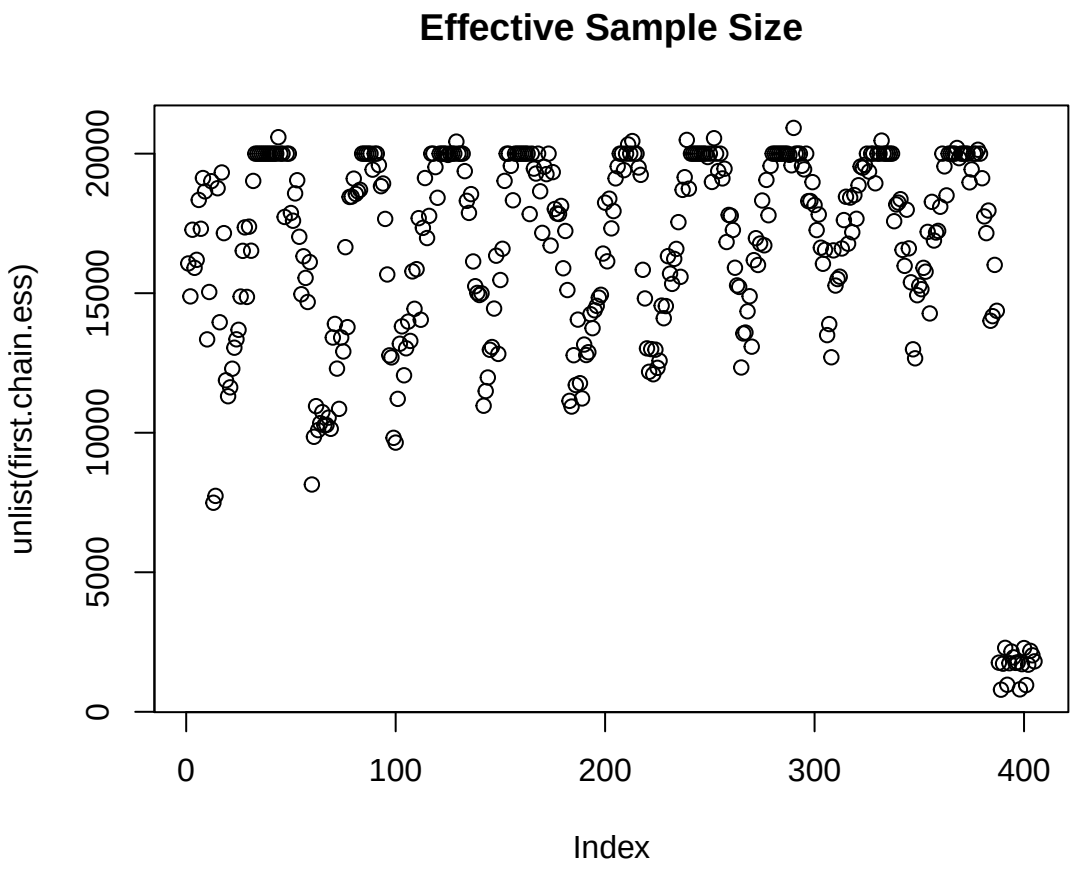
```
# check that our chains length is satisfactory.
#raftery.diag(out.coda) - Indicated 3k so we ran for 40k

pois.geweke <- geweke.diag(out.coda)
#geweke.plot(out.coda)
zs <-pois.geweke[[1]]
z.scores <-unlist (zs['z'])
plot(z.scores, main="Geweke Z-scores for all tracked variables in Poisson GLM")
```




```
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")
```




```
pval.m <- matrix(nrow = 9,ncol = 2)
for(k in 1:9){
  # 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[,paste("Dm[",k,"]",sep='')],chain[,paste("Dsd[",k,"]",sep='')])
  pval1 <- rep(0,2)
  names(pval1)<-Dnames

  for(j in 1:2){
  pval1[j] <- mean(D1[,j]>D0[j])
  }
  pval.m[k,] <- pval1
}
colnames(pval.m)<-c("pval.mean","pval.sd")
rownames(pval.m)<-city_names
pander(data.frame(pval.m), caption = "Baeysian p-values Poisson GLM")
```

Table 3: Baeysian p-values Poisson GLM

	pval.mean	pval.sd
Phoenix	0.4199	0.1532
Denver	0.4911	0.5173
Las Vegas	0.4345	0.1522
Albuquerque	0.4337	0.0147
Tucson	0.5019	0.2683
Salt Lake City	0.4562	0.1472
Los Angeles	0.4199	0.09055
San Francisco	0.4617	0.2398
San Diego	0.4685	0.1419

```
####Predictions Median
predictedMedian <- matrix(nrow = 41,ncol = 9)
diff.pred.train <- matrix(nrow = 41,ncol = 9)
for( i in 1:length(rownames(so$quantiles)) )
{
  rn.so <- rownames(so$quantiles)[i]

  if(grepl("Yp",rn.so)  )
  {
    idx <-gsub('Yp','',rn.so)
    idx <-gsub('\\[','',idx)
    idx<-gsub('\\]','',idx)
    strsplit(idx,",")
    idi <- as.numeric(strsplit(idx,",")[[1]][1])
    idj <- as.numeric(strsplit(idx,",")[[1]][2])
    predictedMedian[idi,idj] <- so$quantiles[i,][3]# 50% Quantiles for predicted
    diff.pred.train[idi,idj] <-  predictedMedian[idi,idj] - X.num[idi,idj]

  }else{
    next
  }
}

train.mse.median <- sum(diff.pred.train^2)/(41*9)

####Predictions Mode - don't need fancy mode fn since it's count data
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
  }

chain <- out.coda[[1]]
predictedMode <- matrix(nrow = 41,ncol = 9)
diff.pred.train.mode <- matrix(nrow = 41,ncol = 9)
for( i in 1:ncol(chain) )
{
  colname <- colnames(chain)[i]
  if(grepl("Yp",colname)  )
  {
    idx <-gsub('Yp','',colname)
```

```
    idx <-gsub('\\[','',idx)
    idx<-gsub('\\]','',idx)
    strsplit(idx,",")
    idi <- as.numeric(strsplit(idx,",")[[1]][1])
    idj <- as.numeric(strsplit(idx,",")[[1]][2])
    samples <- chain[,i]
    predictedMode[idi,idj] <- as.numeric(Mode(samples))
    diff.pred.train.mode[idi,idj] <-  predictedMode[idi,idj] - X.num[idi,idj]

  }else{
    next
  }
}

train.mse.mode <- sum(diff.pred.train.mode^2)/(41*9)

####Predictions Mean
chain <- out.coda[[1]]
predictedMean <- matrix(nrow = 41,ncol = 9)
diff.pred.train.mean <- matrix(nrow = 41,ncol = 9)
for( i in 1:ncol(chain) )
{
  colname <- colnames(chain)[i]
  if(grepl("Yp",colname)  )
  {
    idx <-gsub('Yp','',colname)
    idx <-gsub('\\[','',idx)
    idx<-gsub('\\]','',idx)
    strsplit(idx,",")
    idi <- as.numeric(strsplit(idx,",")[[1]][1])
    idj <- as.numeric(strsplit(idx,",")[[1]][2])
    samples <- chain[,i]
    predictedMean[idi,idj] <- as.numeric(mean(samples))
    diff.pred.train.mean[idi,idj] <-  predictedMean[idi,idj] - X.num[idi,idj]

  }else{
    next
  }
}
train.mse.mean <- sum(diff.pred.train.mean^2)/(41*9)

pois.mse <- data.frame(train.mse.mean=train.mse.mean,train.mse.median=train.mse.median,train.mse.mode=
pander (pois.mse, caption="MSE - via posteriaor mean,median and mode")
```

Table 4: MSE - via posteriaor mean,median and mode

train.mse.mean	train.mse.median	train.mse.mode
1.985	2.152	2.252

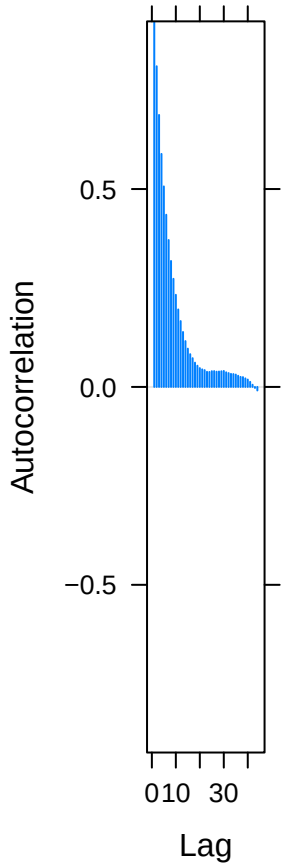
Fit JAGS Negative Binomial Random Effects

```
######################################## Fit JAGS Negative Binomial Random Effects
model_nb = '
model
{
    ## Likelihood
    for(i in 1:N){
      for(j in 1:9){
        Y[i,j] ~ dnegbin(p[i,j],r[j])
        p[i,j] <- r[j]/(r[j]+lambda[i,j])
        log(lambda[i,j]) <- mu[i,j]
        mu[i,j] <- alpha[j] + beta[j]*t[i]
      }
    }

  ## Priors
  for(i in 1:9){
    alpha[i] ~ dnorm(0,taus[i])
    taus[i] ~ dgamma(0.1,0.1)
  }

  # Slopes
  for(i in 1:9){
    beta[i] ~ dnorm(mu.beta,taus.beta[i])
    taus.beta[i] ~ dgamma(0.1,0.1)
  }

  # r
  for(i in 1:9){
    r[i] ~ dunif(0,10)
  }

  ## Posterior Predictive Checks
  for(i in 1:N){
    for(j in 1:9){
        Y2[i,j] ~ dnegbin(p[i,j],r[j])
    }
  }

  for(j in 1:9){
    Dm[j] <- mean(Y2[,j])
    Dsd[j] <- sd(Y2[,j])
  }

  #Prediction
  for(i in 1:N){
    for(j in 1:9){
      Yp[i,j] ~ dnegbin(pp[i,j],r[j])
      pp[i,j] <- r[j]/(r[j]+lambdap[i,j])
      log(lambdap[i,j]) <- mup[i,j]
      mup[i,j] <- alpha[j] + beta[j]*t[i]
    }
```

```
  }
}
'
  # Set up the data
  model_data = list(N = 41, t=seq(1:41),Y=X.num,mu.beta=0,tau.beta=.0001,mu.intercept=0,tau.intercept=.
  # Choose the parameters to watch
  model_parameters =  c("r","beta", "alpha","Dm","Dsd","Yp")# model_parameters =  c("r")
  model_nb <- jags.model(textConnection(model_nb),data = model_data,n.chains = n.chains)#Compile Model
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 369
##    Unobserved stochastic nodes: 783
##    Total graph size: 3070
##
## Initializing model
```

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

  coda::acfplot(out.coda[[1]][,'beta[1]'],100)
```




```
#coda::crosscorr.plot(out.coda[[1]][,'p[1,1]'],out.coda[[1]][,'r[1]'])
#coda::crosscorr.plot(out.coda[[1]])

slopes.hpd <- matrix(nrow = 9,ncol=2)
for(k in 1:9){
  coef.name <- paste('beta[', k,']', sep='')
  inv <- HPDinterval(out.coda[[1]][,coef.name],.95)
  slopes.hpd[k,]<-inv
}
colnames(slopes.hpd) <- c("lower","upper")
rownames(slopes.hpd) <-city_names
pander(data.frame(slopes.hpd), caption = "0.95 HPD Intervals for slopes")
```

Table 5: 0.95 HPD Intervals for slopes

	lower	upper
Phoenix	-0.03486	0.0152
Denver	0.006148	0.08619
Las Vegas	-0.01128	0.03479
Albuquerque	-0.05797	0.002841
Tucson	0.01975	0.0722
Salt Lake City	-0.009478	0.03619
Los Angeles	-0.03448	0.01763

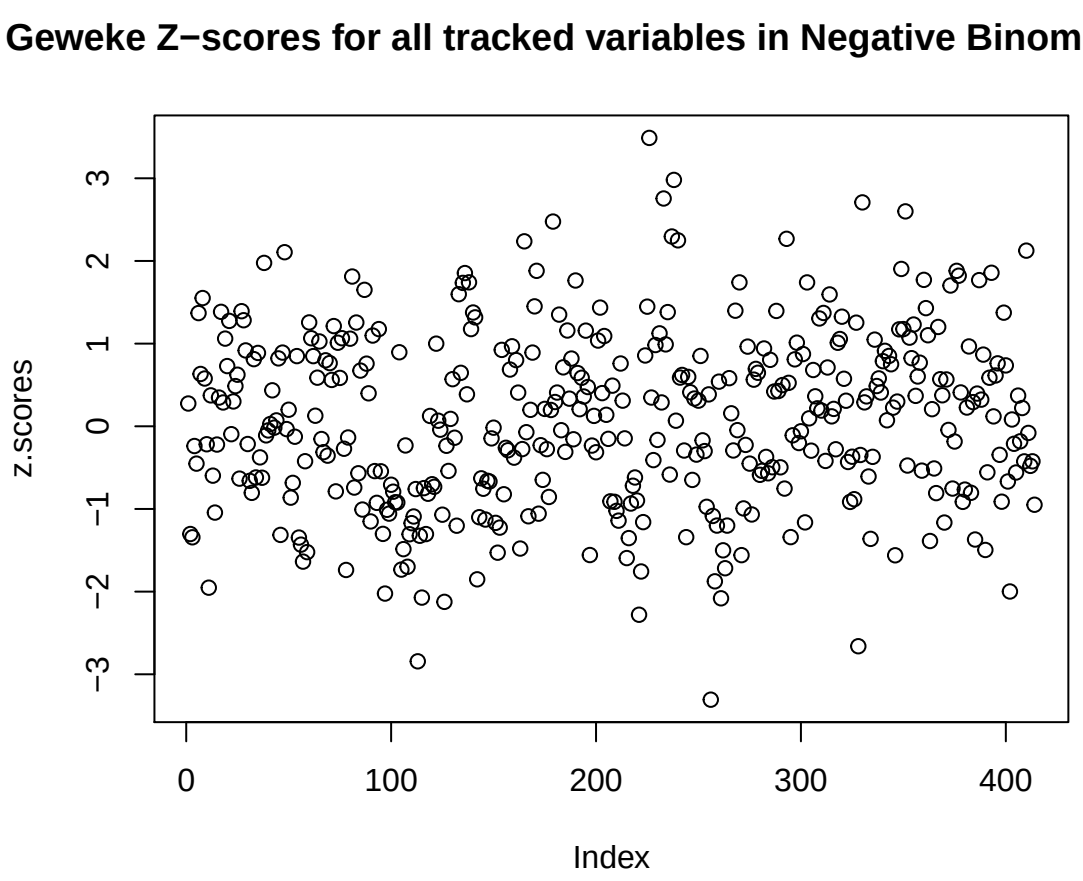
	lower	upper
San Francisco	-0.01943	0.03235
San Diego	-0.006653	0.04356

```
#assess the posteriors stationarity, by looking at the Heidelberg-Welch convergence diagnostic:
hd <- heidel.diag(out.coda)
hdd <- hd[[1]]
hd.pass <- hdd[,2]
hd.fail <-sum(is.na(hd.pass))
pander(data.frame(fail.count = hd.fail), caption ="Fail count for Heidelberger and Welch diagnotic")
```

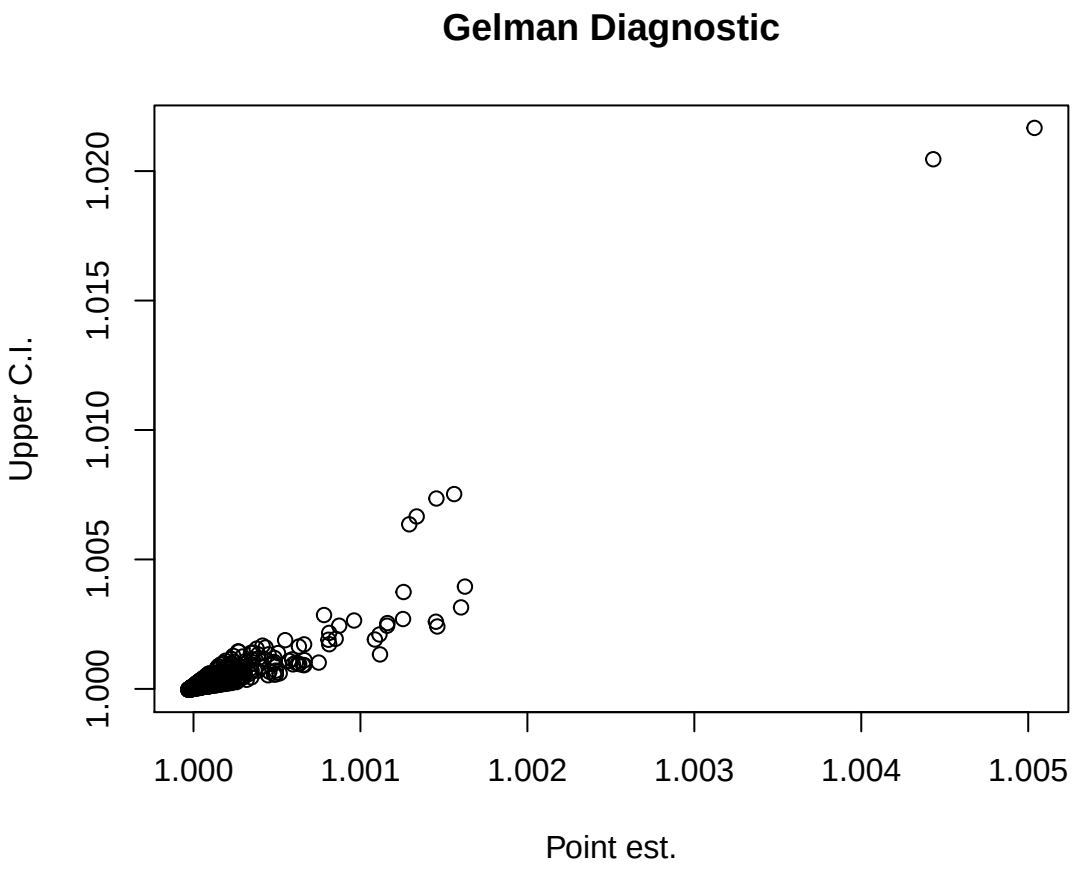
Table 6: Fail count for Heidelberger and Welch diagnotic

fail.count
1

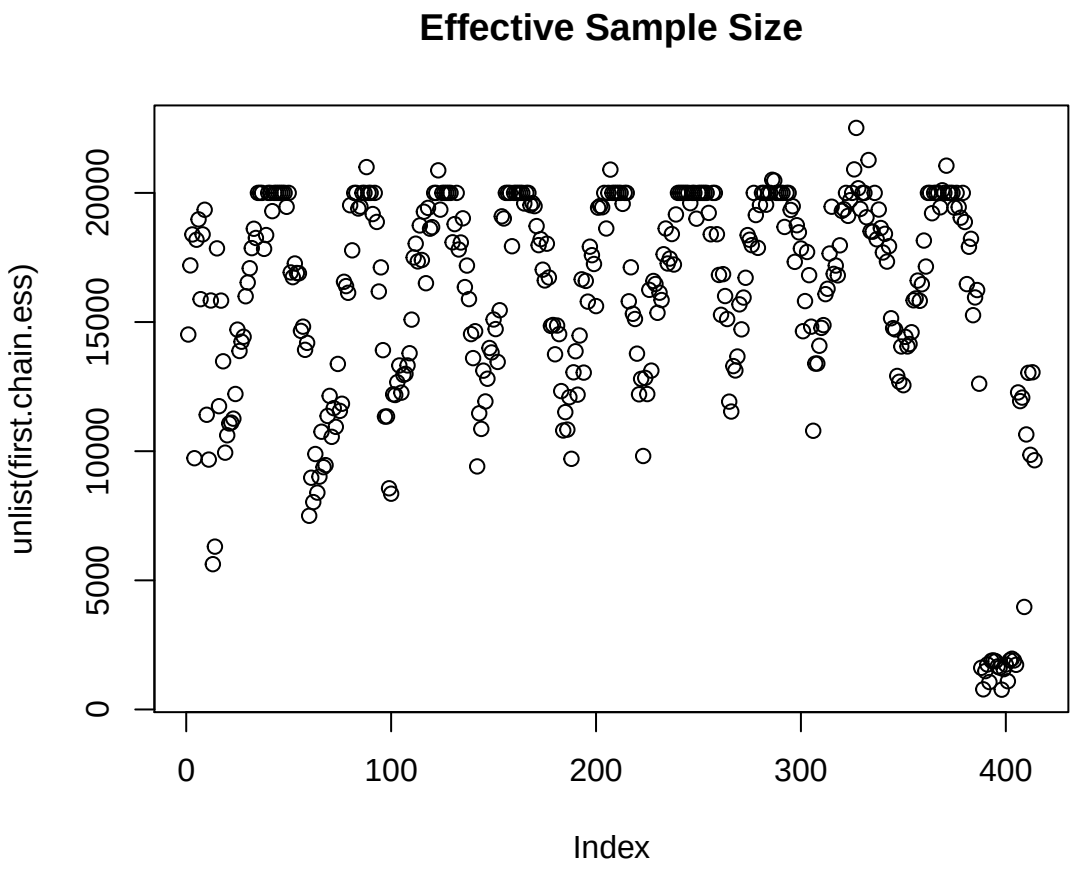
```
# check that our chains length is satisfactory.
#raftery.diag(out.coda) - Indicated 3k so we ran for 40k

nb.geweke <- geweke.diag(out.coda)
#geweke.plot(out.coda)
zs <-nb.geweke[[1]]
z.scores <-unlist (zs['z'])
plot(z.scores, main="Geweke Z-scores for all tracked variables in Negative Binomial GLM")
```




```
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")
```




```
pval.m <- matrix(nrow = 9,ncol = 2)
 for(k in 1:9){
   # 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[,paste("Dm[",k,"]",sep='')],chain[,paste("Dsd[",k,"]",sep='')])
   pval1 <- rep(0,2)
   names(pval1)<-Dnames

   for(j in 1:2){
   pval1[j] <- mean(D1[,j]>D0[j])
   }
   pval.m[k,] <- pval1
 }
 colnames(pval.m)<-c("pval.mean","pval.sd")
 pander(data.frame(pval.m), caption = "Baeysian p-values Poisson GLM")
```

Table 7: Baeysian p-values Poisson GLM

pval.mean	pval.sd
0.437	0.477

pval.mean	pval.sd
0.5089	0.6384
0.4505	0.5201
0.4456	0.4031
0.5367	0.6001
0.4679	0.431
0.4361	0.3908
0.4819	0.4954
0.4844	0.469

```
####Predictions Median
predictedMedian <- matrix(nrow = 41,ncol = 9)
diff.pred.train <- matrix(nrow = 41,ncol = 9)
for( i in 1:length(rownames(so$quantiles)) )
{
  rn.so <- rownames(so$quantiles)[i]

  if(grepl("Yp",rn.so)  )
  {
    idx <-gsub('Yp','',rn.so)
    idx <-gsub('\\[','',idx)
    idx<-gsub('\\]','',idx)
    strsplit(idx,",")
    idi <- as.numeric(strsplit(idx,",")[[1]][1])
    idj <- as.numeric(strsplit(idx,",")[[1]][2])
    predictedMedian[idi,idj] <- so$quantiles[i,][3]# 50% Quantiles for predicted
    diff.pred.train[idi,idj] <-  predictedMedian[idi,idj] - X.num[idi,idj]

  }else{
    next
  }
}

train.mse.median <- sum(diff.pred.train^2)/(41*9)

####Predictions Mode - don't need fancy mode fn since it's count data
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
  }

chain <- out.coda[[1]]
predictedMode <- matrix(nrow = 41,ncol = 9)
diff.pred.train.mode <- matrix(nrow = 41,ncol = 9)
for( i in 1:ncol(chain) )
{
  colname <- colnames(chain)[i]
  if(grepl("Yp",colname)  )
  {
    idx <-gsub('Yp','',colname)
    idx <-gsub('\\[','',idx)
    idx<-gsub('\\]','',idx)
    strsplit(idx,",")
```

```
    idi <- as.numeric(strsplit(idx,",")[[1]][1])
    idj <- as.numeric(strsplit(idx,",")[[1]][2])
    samples <- chain[,i]
    predictedMode[idi,idj] <- as.numeric(Mode(samples))
    diff.pred.train.mode[idi,idj] <-  predictedMode[idi,idj] - X.num[idi,idj]

  }else{
    next
  }
}

train.mse.mode <- sum(diff.pred.train.mode^2)/(41*9)

####Predictions Mean
chain <- out.coda[[1]]
predictedMean <- matrix(nrow = 41,ncol = 9)
diff.pred.train.mean <- matrix(nrow = 41,ncol = 9)
for( i in 1:ncol(chain) )
{
  colname <- colnames(chain)[i]
  if(grepl("Yp",colname)  )
  {
    idx <-gsub('Yp','',colname)
    idx <-gsub('\\[','',idx)
    idx<-gsub('\\]','',idx)
    strsplit(idx,",")
    idi <- as.numeric(strsplit(idx,",")[[1]][1])
    idj <- as.numeric(strsplit(idx,",")[[1]][2])
    samples <- chain[,i]
    predictedMean[idi,idj] <- as.numeric(mean(samples))
    diff.pred.train.mean[idi,idj] <-  predictedMean[idi,idj] - X.num[idi,idj]

  }else{
    next
  }
}
train.mse.mean <- sum(diff.pred.train.mean^2)/(41*9)

nb.mse <- data.frame(train.mse.mean=train.mse.mean,train.mse.median=train.mse.median,train.mse.mode=t
pander (nb.mse, caption="MSE - via posteriaor mean,median and mode")
```

Table 8: MSE - via posteriaor mean,median and mode

train.mse.mean	train.mse.median	train.mse.mode
1.999	2.152	3.165

Fit JAGS Poisson GLM With Lattitude

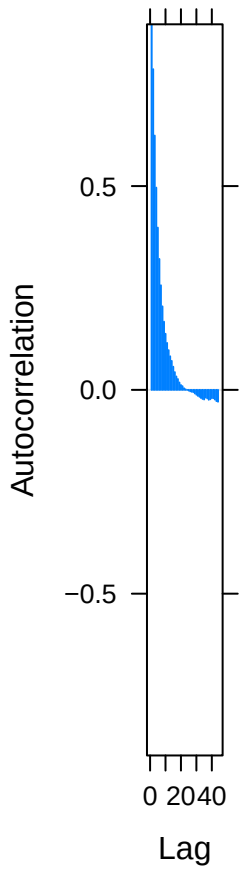
```
lattitude<- c(33.4484,39.7392,36.1699,35.0844,32.2226,40.7608,34.0522,37.7749,32.7157)
lattitude<- as.vector(scale(lattitude))
```

```
######################################## Fit JAGS Poisson
model_pois = '
model
{
    ## Likelihood
    for(i in 1:N){
      for(j in 1:9){
        Y[i,j] ~ dpois(lambda[i,j])
        log(lambda[i,j]) <- mu[i,j]
        mu[i,j] <- alpha[j] + beta[j]*t[i] +gamma[j]*lattitude[j]
      }
    }

  ## Priors
  for(i in 1:9){
    alpha[i] ~ dnorm(0,taus[i])
    taus[i] ~ dgamma(0.1,0.1)
  }


  for(i in 1:9){
    gamma[i] ~ dnorm(0,taus.gamma[i])
    taus.gamma[i] ~ dgamma(0.1,0.1)
  }

  # Slopes
  for(i in 1:9){
    beta[i] ~ dnorm(0,taus.beta[i])
    taus.beta[i] ~ dgamma(0.1,0.1)
  }

  ## Posterior Predictive Checks
  for(i in 1:N){
    for(j in 1:9){
        Y2[i,j] ~ dpois(lambda[i,j])
    }
  }

  for(j in 1:9){
    Dm[j] <- mean(Y2[,j])
    Dsd[j] <- sd(Y2[,j])
  }

  #Prediction
  for(i in 1:N){
    for(j in 1:9){
      Yp[i,j] ~ dpois(lambdap[i,j])
      log(lambdap[i,j]) <- mup[i,j]
      mup[i,j] <- alpha[j] + beta[j]*t[i] +gamma[j]*lattitude[j]
    }
  }
}
'
```

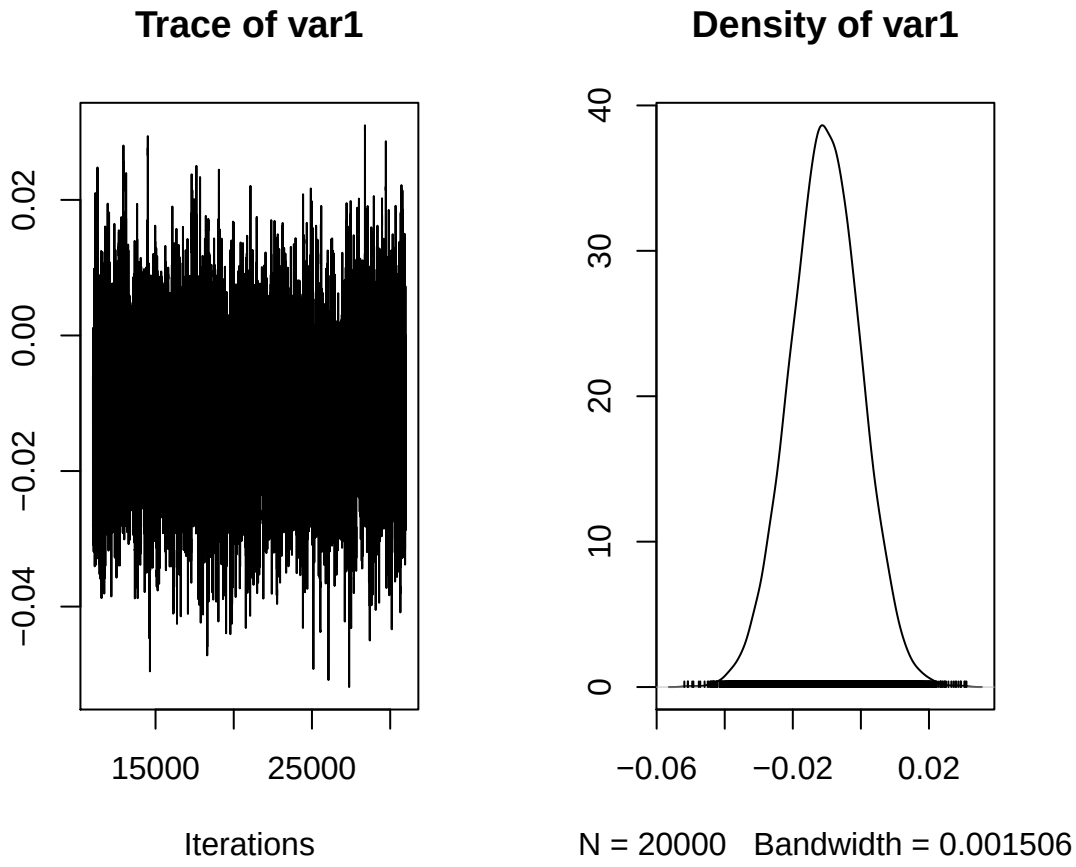
```r
# Set up the data
model_data = list(N = 41, t=seq(1:41),Y=X.num,lattitude=lattitude  )
# Choose the parameters to watch
model_parameters =  c("beta", "alpha","gamma","Dm", "Dsd", "Yp")
model_pois <- jags.model(textConnection(model_pois),data = model_data,n.chains = n.chains)#Compile Mo
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 369
##    Unobserved stochastic nodes: 792
##    Total graph size: 2357
##
## Initializing model
```

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

coda::acfplot(out.coda[[1]][,'beta[1]'],100)
```



```
plot(out.coda[[1]][,'beta[1]'])
```




```
slopes.hpd <- matrix(nrow = 9,ncol=2)
for(k in 1:9){
  coef.name <- paste('beta[', k,']', sep='')
  inv <- HPDinterval(out.coda[[1]][,coef.name],.95)
  slopes.hpd[k,]<-inv
}
colnames(slopes.hpd) <- c("lower","upper")
rownames(slopes.hpd) <-city_names
pander(data.frame(slopes.hpd), caption = "0.95 HPD Intervals for slopes")
```

Table 9: 0.95 HPD Intervals for slopes

	lower	upper
Phoenix	-0.03107	0.009616
Denver	0.01223	0.08157
Las Vegas	-0.009938	0.02797
Albuquerque	-0.04773	-0.007609
Tucson	0.02089	0.06453
Salt Lake City	-0.01083	0.03327
Los Angeles	-0.03382	0.01177
San Francisco	-0.01734	0.0297

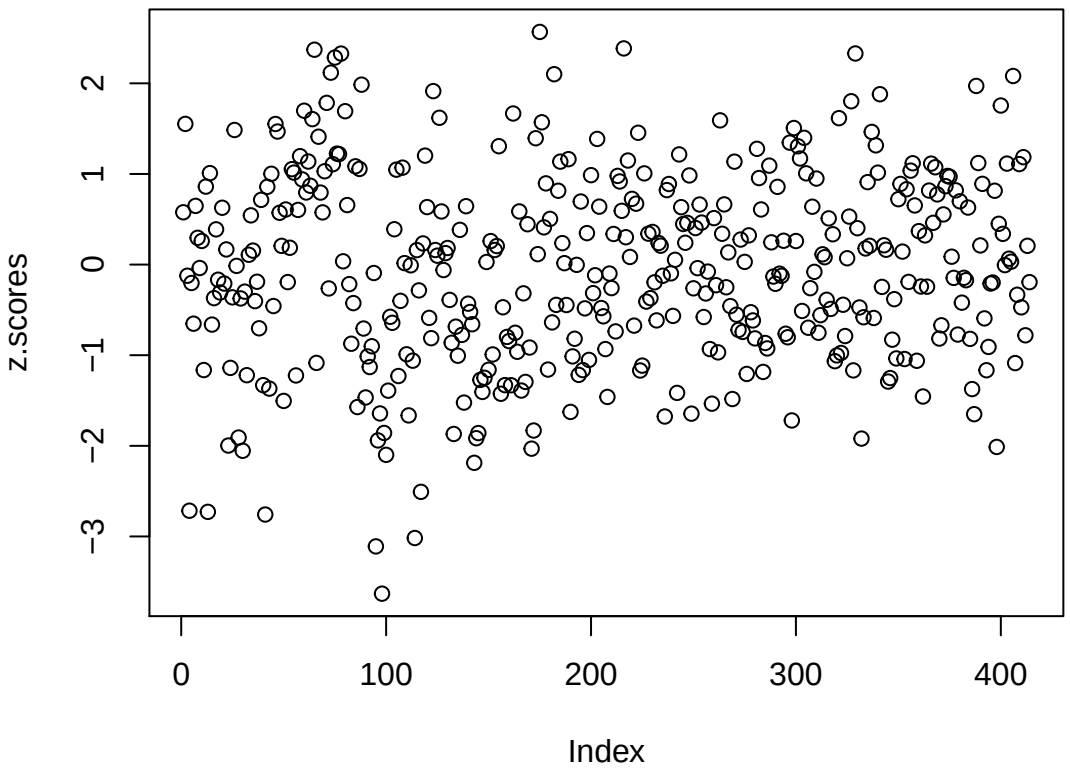
	lower	upper
San Diego	-0.005584	0.03822

```
so <-summary(out.coda)

#assess the posteriors stationarity, by looking at the Heidelberg-Welch convergence diagnostic:
hd <- heidel.diag(out.coda)
hdd <- hd[[1]]
hd.pass <- hdd[,2]
hd.fail <-sum(is.na(hd.pass))
pander(data.frame(fail.count = hd.fail), caption ="Fail count for Heidelberger and Welch diagnotic")
```

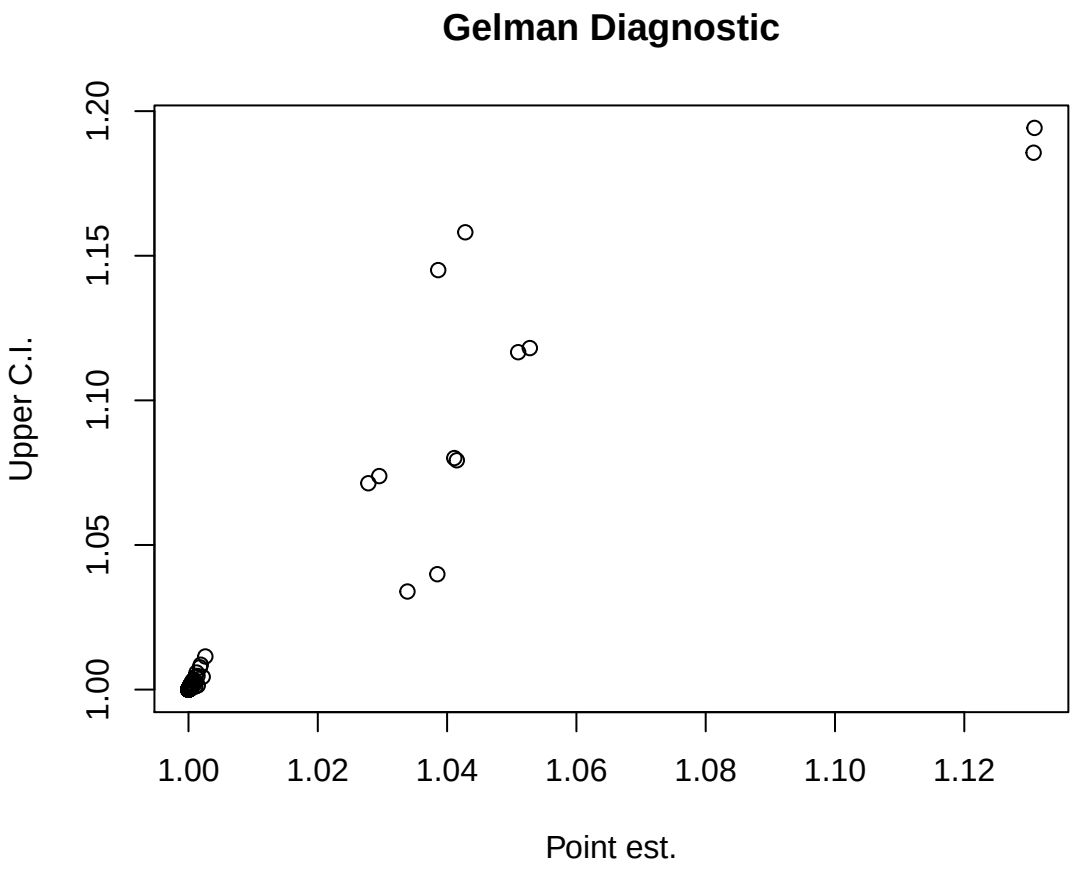
Table 10: Fail count for Heidelberger and Welch diagnotic

fail.count
9

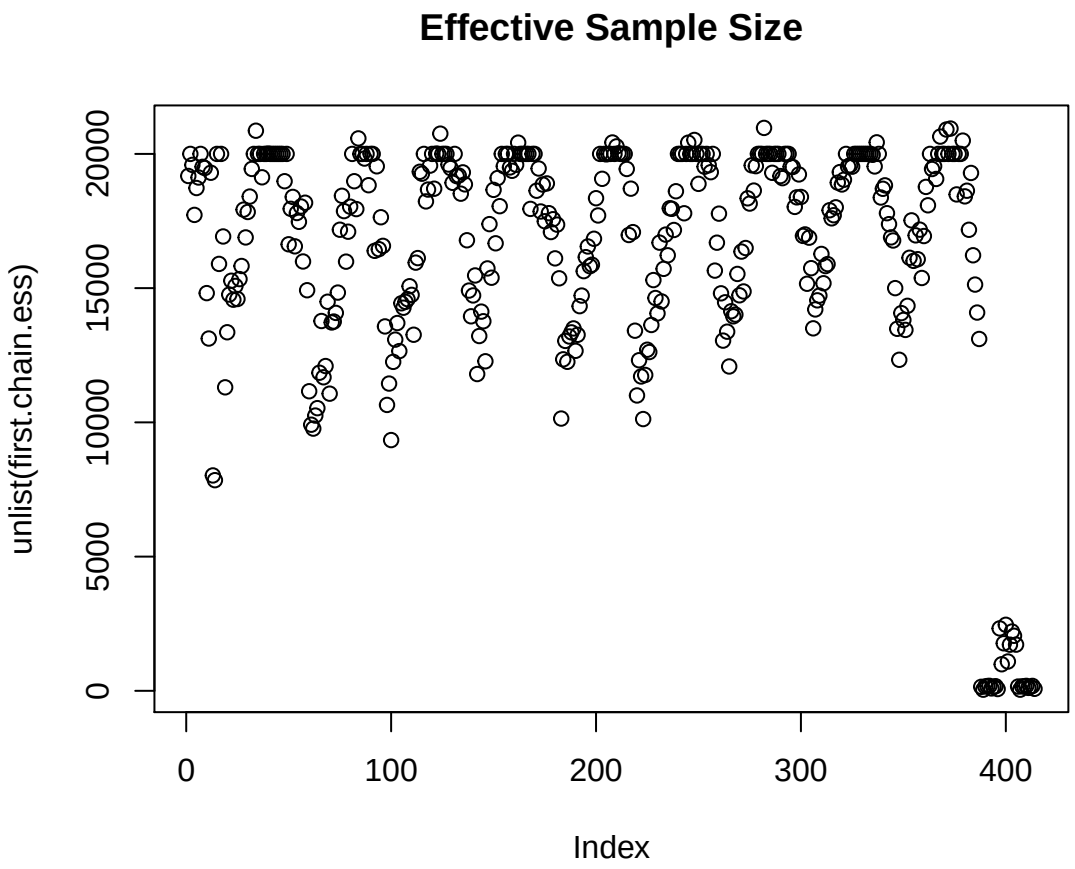
```
# check that our chains length is satisfactory.
#raftery.diag(out.coda) - Indicated 3k so we ran for 40k

pois.geweke <- geweke.diag(out.coda)
#geweke.plot(out.coda)
zs <-pois.geweke[[1]]
z.scores <-unlist (zs['z'])
plot(z.scores, main="Geweke Z-scores for all tracked variables in Poisson GLM")
```




```
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")
```




```
pval.m <- matrix(nrow = 9,ncol = 2)
for(k in 1:9){
  # 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[,paste("Dm[",k,"]",sep='')],chain[,paste("Dsd[",k,"]",sep='')])
  pval1 <- rep(0,2)
  names(pval1)<-Dnames

  for(j in 1:2){
  pval1[j] <- mean(D1[,j]>D0[j])
  }
  pval.m[k,] <- pval1
}
colnames(pval.m)<-c("pval.mean","pval.sd")
pander(data.frame(pval.m), caption = "Baeysian p-values Poisson GLM")
```

Table 11: Baeysian p-values Poisson GLM

pval.mean	pval.sd
0.448	0.1681

pval.mean	pval.sd
0.4676	0.5139
0.4436	0.1518
0.4422	0.01405
0.4798	0.2848
0.4659	0.148
0.4466	0.1031
0.4584	0.2427
0.4663	0.1454

```
####Predictions Median
predictedMedian <- matrix(nrow = 41,ncol = 9)
diff.pred.train <- matrix(nrow = 41,ncol = 9)
for( i in 1:length(rownames(so$quantiles)) )
{
  rn.so <- rownames(so$quantiles)[i]

  if(grepl("Yp",rn.so)  )
  {
    idx <-gsub('Yp','',rn.so)
    idx <-gsub('\\[','',idx)
    idx<-gsub('\\]','',idx)
    strsplit(idx,",")
    idi <- as.numeric(strsplit(idx,",")[[1]][1])
    idj <- as.numeric(strsplit(idx,",")[[1]][2])
    predictedMedian[idi,idj] <- so$quantiles[i,][3]# 50% Quantiles for predicted
    diff.pred.train[idi,idj] <-  predictedMedian[idi,idj] - X.num[idi,idj]

  }else{
    next
  }
}

train.mse.median <- sum(diff.pred.train^2)/(41*9)

####Predictions Mode - don't need fancy mode fn since it's count data
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
  }

chain <- out.coda[[1]]
predictedMode <- matrix(nrow = 41,ncol = 9)
diff.pred.train.mode <- matrix(nrow = 41,ncol = 9)
for( i in 1:ncol(chain) )
{
  colname <- colnames(chain)[i]
  if(grepl("Yp",colname)  )
  {
    idx <-gsub('Yp','',colname)
    idx <-gsub('\\[','',idx)
    idx<-gsub('\\]','',idx)
    strsplit(idx,",")
```

```
      idi <- as.numeric(strsplit(idx,",")[[1]][1])
      idj <- as.numeric(strsplit(idx,",")[[1]][2])
      samples <- chain[,i]
      predictedMode[idi,idj] <- as.numeric(Mode(samples))
      diff.pred.train.mode[idi,idj] <-  predictedMode[idi,idj] - X.num[idi,idj]

    }else{
      next
    }
  }

  train.mse.mode <- sum(diff.pred.train.mode^2)/(41*9)

  ####Predictions Mean
  chain <- out.coda[[1]]
  predictedMean <- matrix(nrow = 41,ncol = 9)
  diff.pred.train.mean <- matrix(nrow = 41,ncol = 9)
  for( i in 1:ncol(chain) )
  {
    colname <- colnames(chain)[i]
    if(grepl("Yp",colname)  )
    {
      idx <-gsub('Yp','',colname)
      idx <-gsub('\\[','',idx)
      idx<-gsub('\\]','',idx)
      strsplit(idx,",")
      idi <- as.numeric(strsplit(idx,",")[[1]][1])
      idj <- as.numeric(strsplit(idx,",")[[1]][2])
      samples <- chain[,i]
      predictedMean[idi,idj] <- as.numeric(mean(samples))
      diff.pred.train.mean[idi,idj] <-  predictedMean[idi,idj] - X.num[idi,idj]

    }else{
      next
    }
  }
  train.mse.mean <- sum(diff.pred.train.mean^2)/(41*9)

  pois.mse <- data.frame(train.mse.mean=train.mse.mean,train.mse.median=train.mse.median,train.mse.mode=
  pander (pois.mse, caption="MSE - via posteriaor mean,median and mode")
```

Table 12: MSE - via posteriaor mean,median and mode

train.mse.mean	train.mse.median	train.mse.mode
1.987	2.125	2.241

DIC Calculation

```
dic_pois <- dic.samples(model_pois, variable.names = c("beta",
    "alpha"), n.iter = nSamples, progress.bar = "none")
```

```
dic_pois
```

```
## Mean deviance:  1178
## penalty 17.77
## Penalized deviance: 1196
```

```
dic_nb <- dic.samples(model_nb, variable.names = c("beta",
    "alpha"), n.iter = nSamples, progress.bar = "none")
dic_nb
```

```
## Mean deviance:  1157
## penalty 22.52
## Penalized deviance: 1180
```