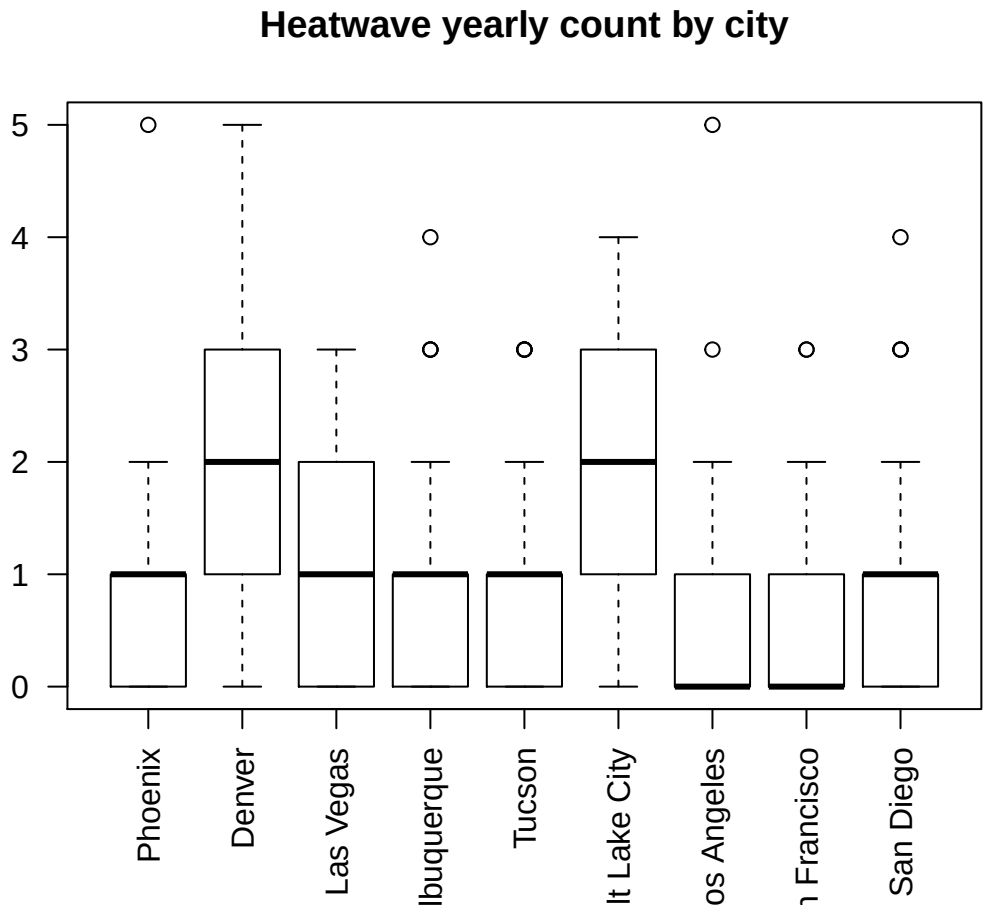
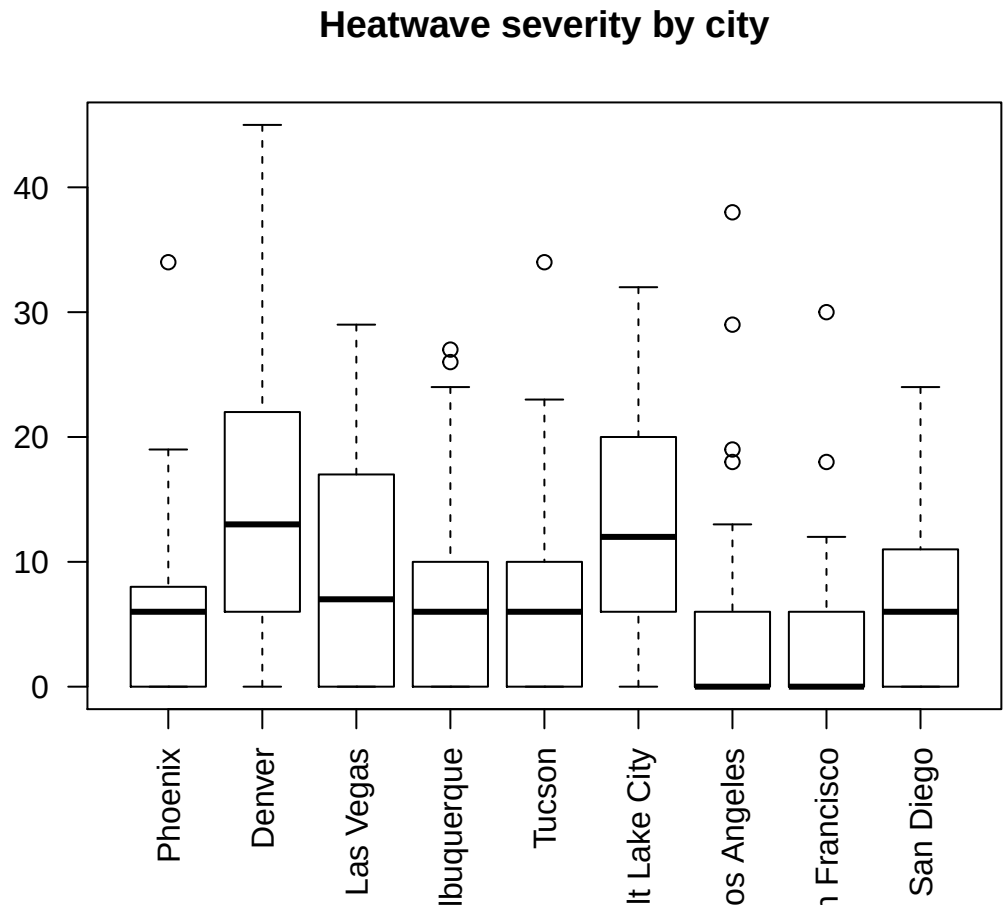
E3

```
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("HWD2.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")
```



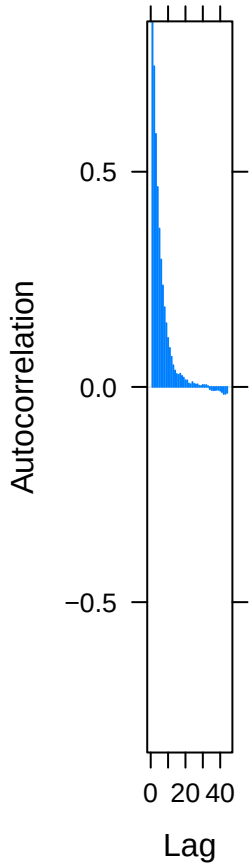
Fit JAGS Poisson Random Effects

```
######################################## 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)
  }
```

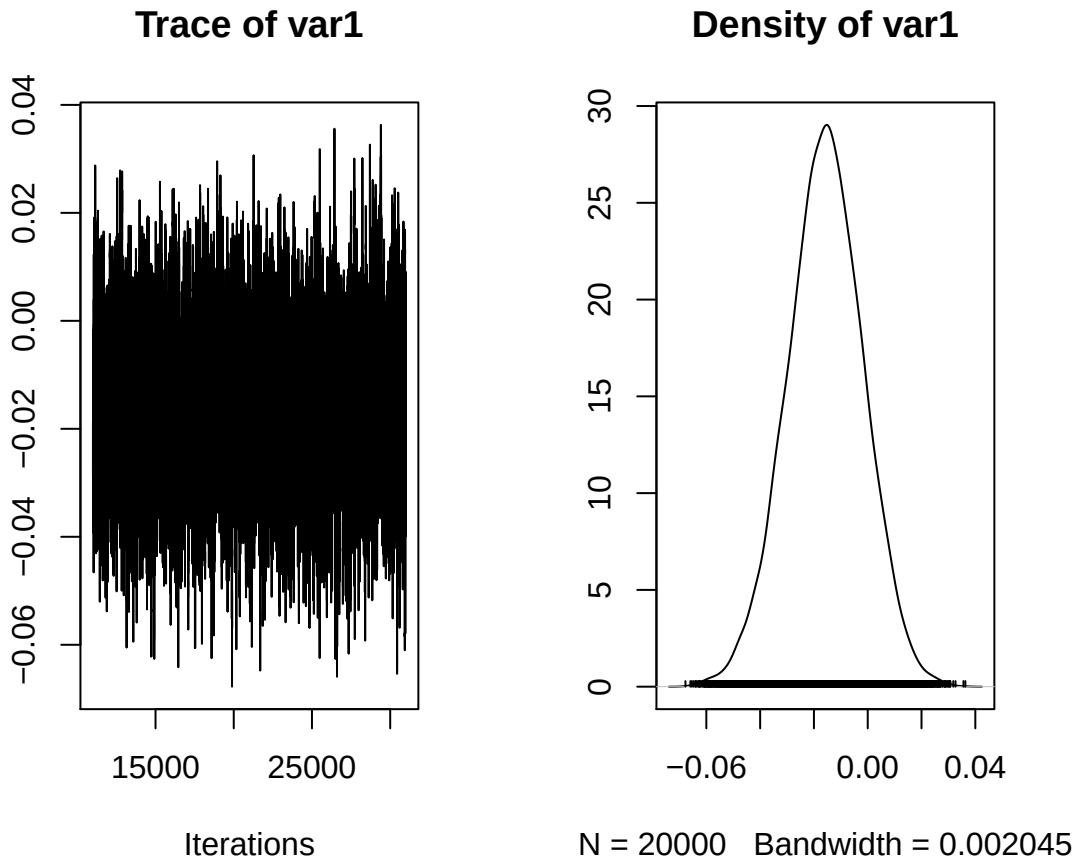
```
  # Slopes
  for(i in 1:9){
    beta[i] ~ dnorm(mu.beta,taus.beta[i])
    taus.beta[i] ~ dgamma(0.1,0.1)
  }

  ## 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

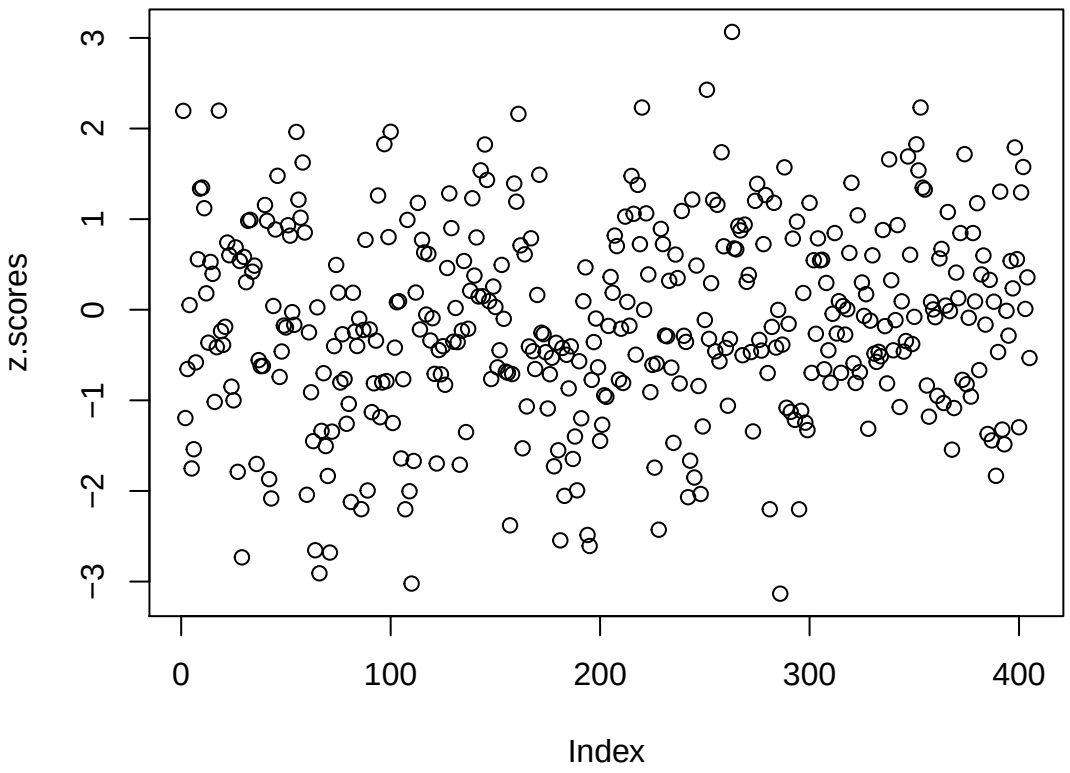
	lower	upper
Phoenix	-0.04299	0.01244
Denver	0.01096	0.04569
Las Vegas	-0.01425	0.03237
Albuquerque	-0.05148	0.002659
Tucson	0.01468	0.07759
Salt Lake City	-0.00499	0.03227
Los Angeles	-0.05611	0.004309
San Francisco	-0.03006	0.04284
San Diego	-0.01449	0.03818

```
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")
```

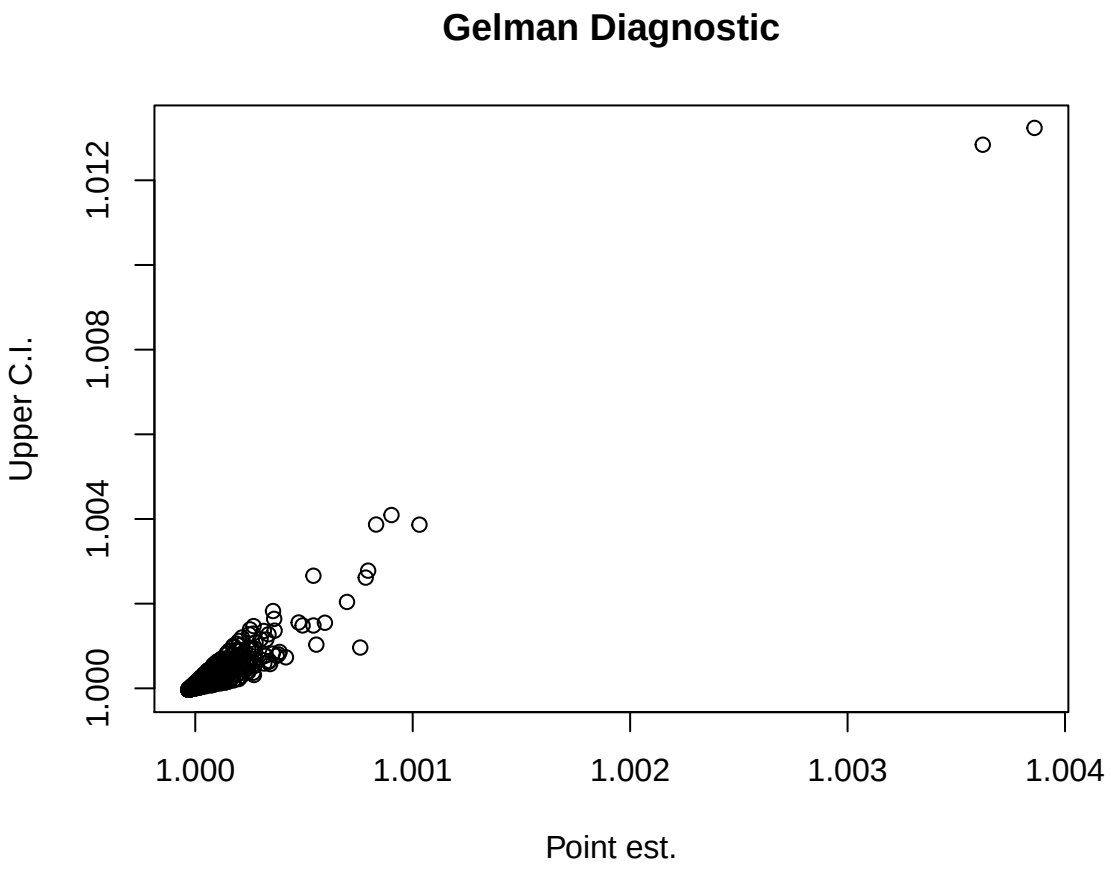
Table 2: Fail count for Heidelberger and Welch diagnotic

fail.count
10

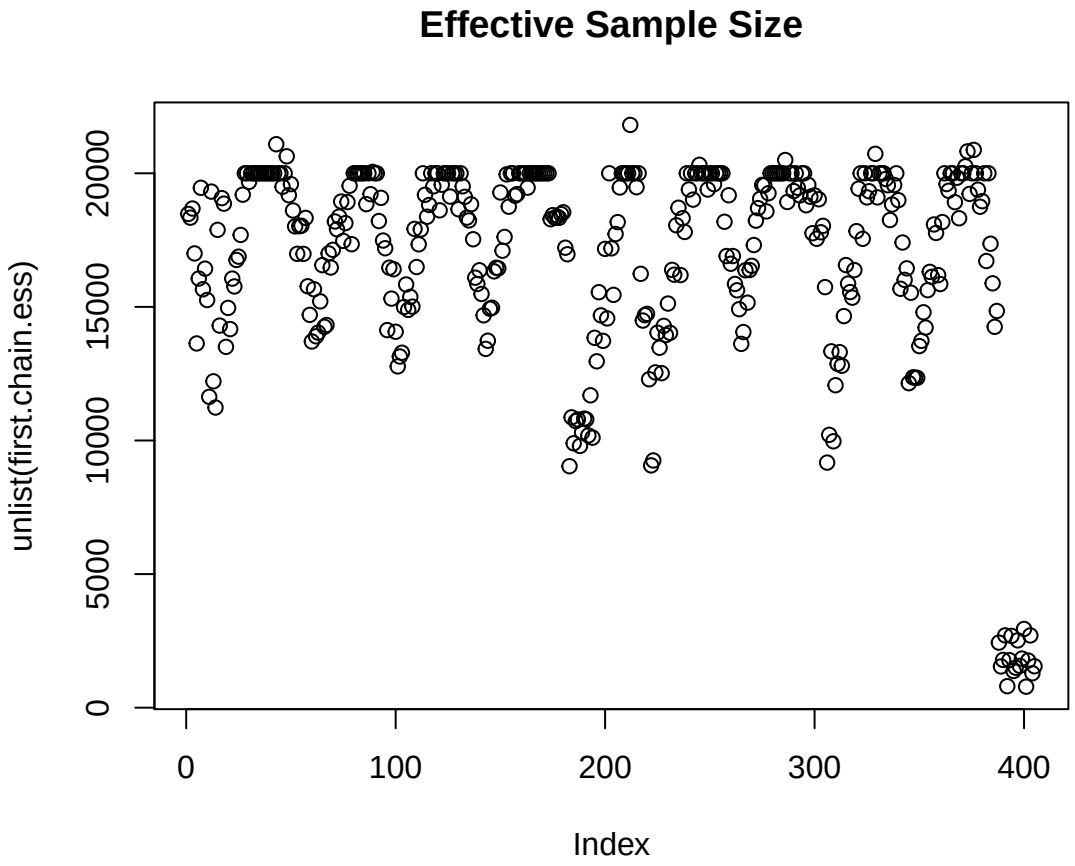
```
# check that our chains length is satisfactory.
#raftery.diag(out.coda) - Indicated 3k so we ran for 40k

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	0.4509	0.266
Denver	0.4671	0.4155
Las Vegas	0.4818	0.3851
Albuquerque	0.4157	0.2161
Tucson	0.4918	0.6603
Salt Lake City	0.442	0.5911
Los Angeles	0.4428	0.0949
San Francisco	0.497	0.2884
San Diego	0.4964	0.2084

```
####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
1.109	1.171	1.545

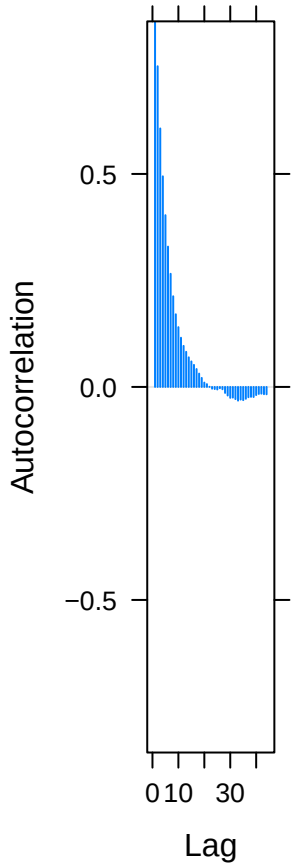
Fit JAGS Negative Binomial Random Effects

```
######################################## 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	-0.04714	0.01487
Denver	0.007428	0.04888
Las Vegas	-0.01717	0.03588
Albuquerque	-0.05389	0.005381
Tucson	0.01517	0.08148
Salt Lake City	-0.006243	0.03605
Los Angeles	-0.05971	0.008228

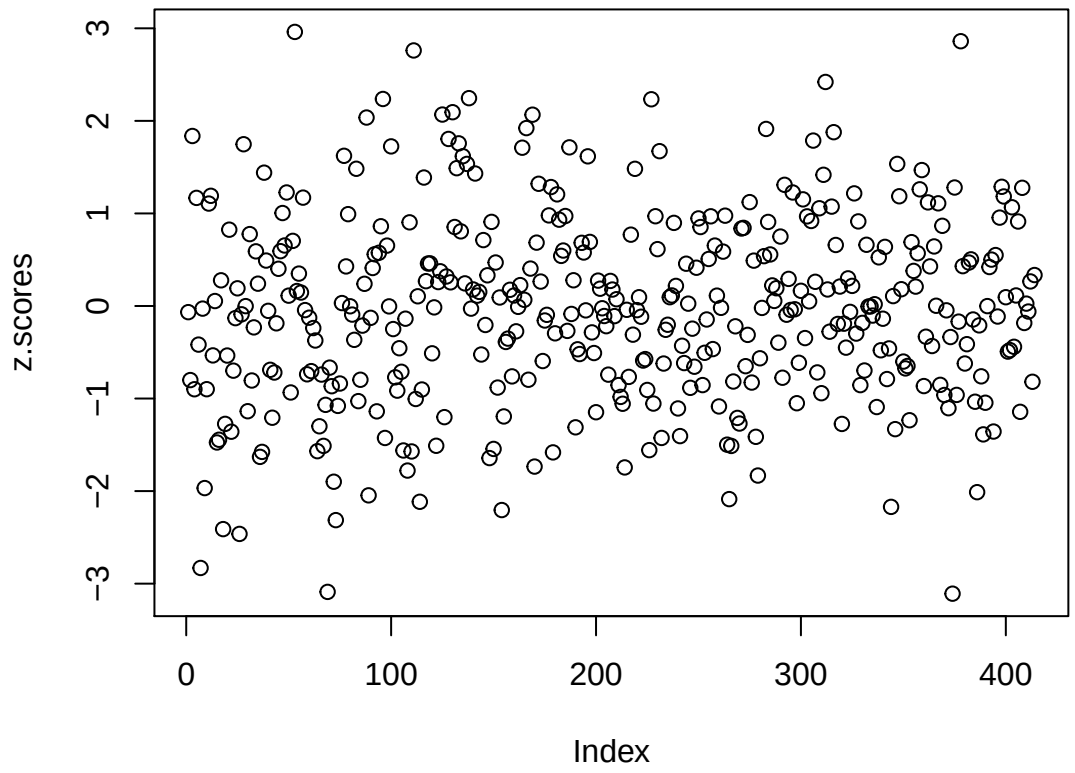
	lower	upper
San Francisco	-0.03319	0.04772
San Diego	-0.01666	0.04233

```
#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")
```

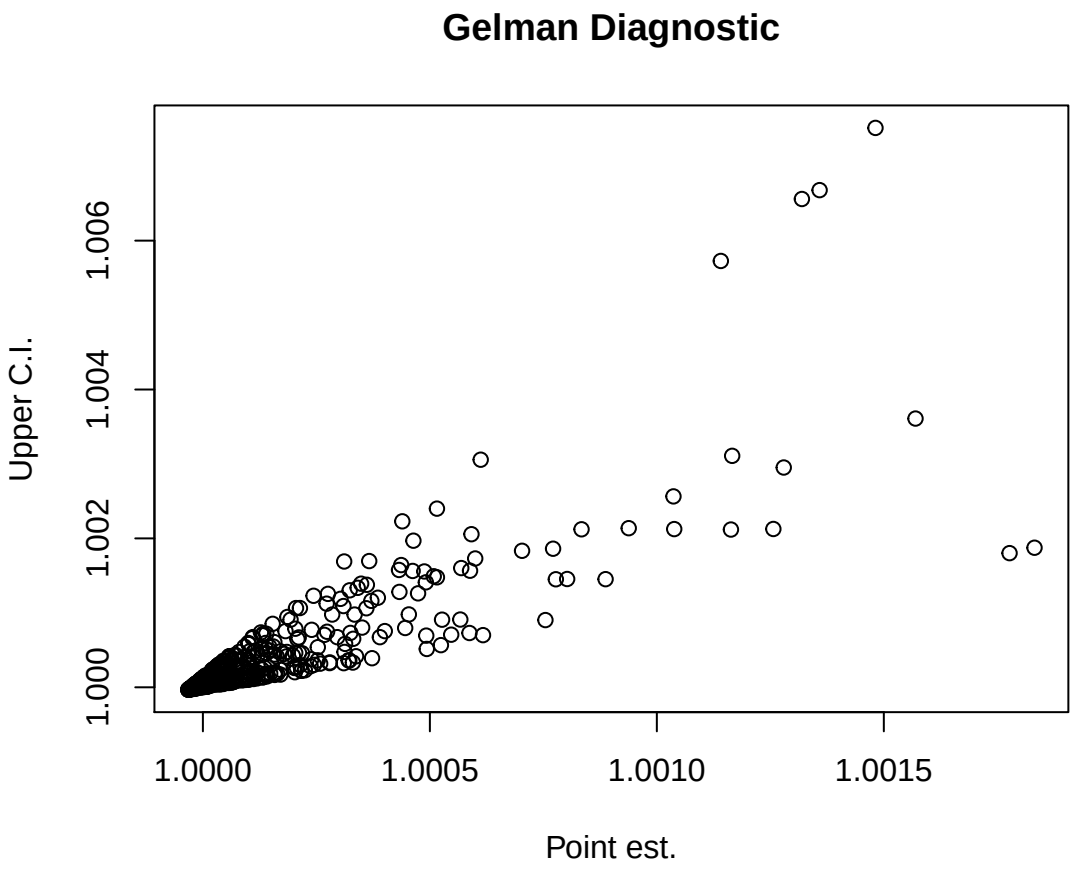
Table 6: Fail count for Heidelberger and Welch diagnotic

fail.count
4

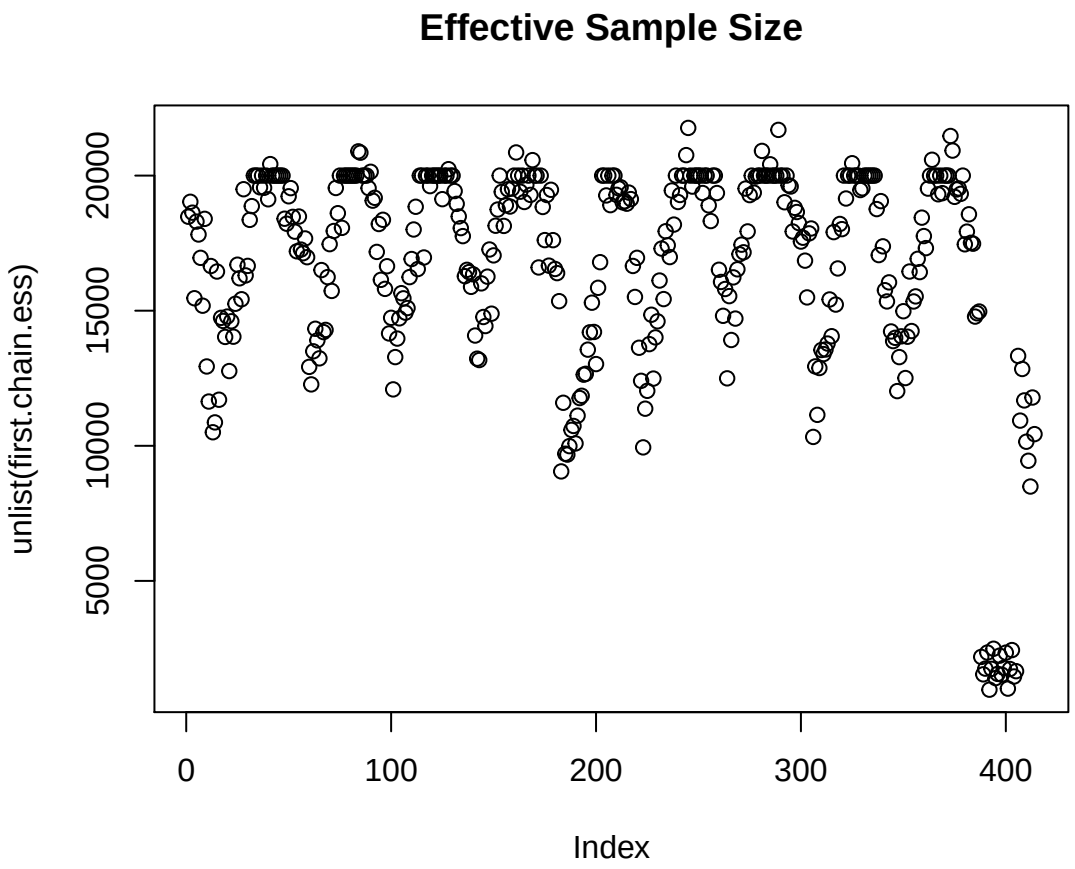
```
# check that our chains length is satisfactory.
#raftery.diag(out.coda) - Indicated 3k so we ran for 40k

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
0.4623	0.4321

pval.mean	pval.sd
0.4928	0.6974
0.4989	0.6126
0.4274	0.4182
0.5051	0.7593
0.4497	0.7988
0.4547	0.2761
0.5161	0.4337
0.5141	0.4452

```
####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
1.113	1.171	1.743

Fit JAGS Poisson GLM With Lattitude

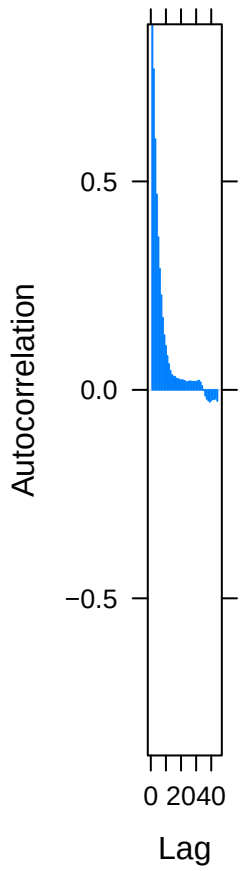
```
lattitude<- c(33.4484,39.7392,36.1699,35.0844,32.2226,40.7608,34.0522,37.7749,32.7157)
lattitude<- as.vector(scale(lattitude))
```

```
######################################## 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]
    }
  }
}
'
```

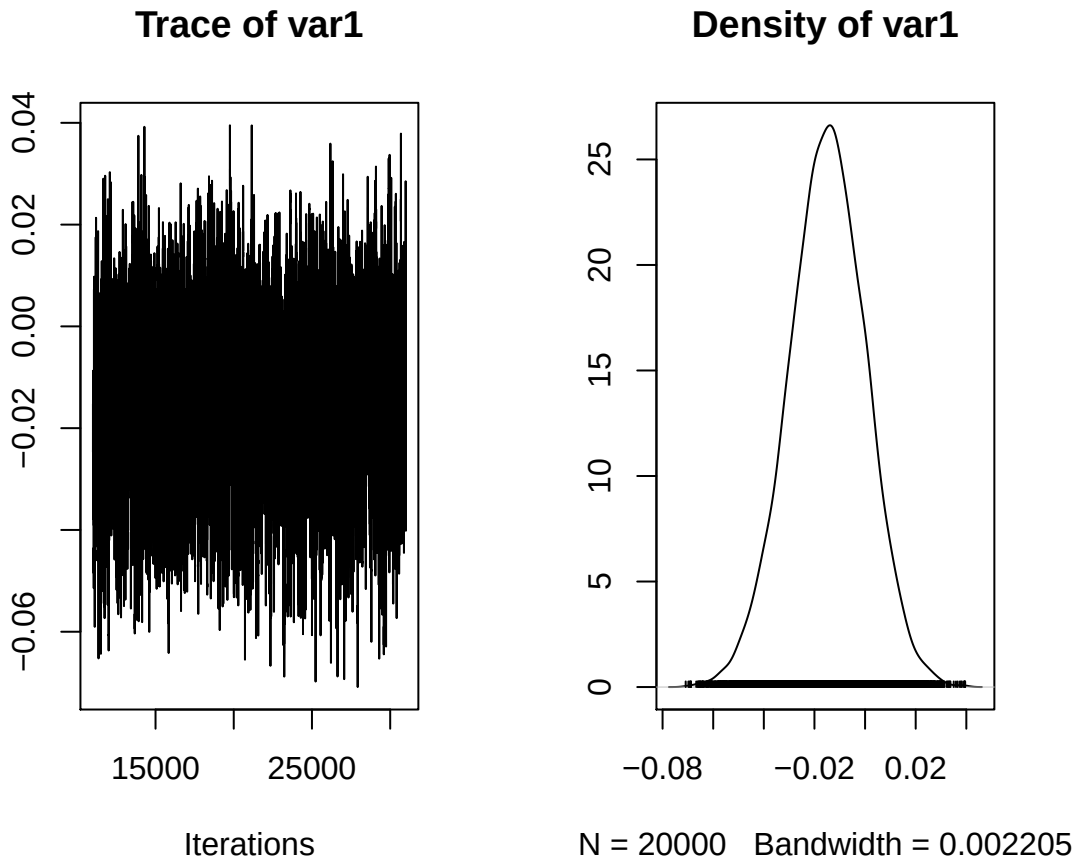
```r
# Set up the data
model_data = list(N = 41, t=seq(1:41),Y=X.num,lattitude=lattitude  )
# Choose the parameters to watch
model_parameters =  c("beta", "alpha","gamma","Dm", "Dsd", "Yp")
model_pois <- jags.model(textConnection(model_pois),data = model_data,n.chains = n.chains)#Compile Mo
```

```
## 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	-0.04513	0.01443
Denver	0.008741	0.04605
Las Vegas	-0.01347	0.03438
Albuquerque	-0.05261	0.002738
Tucson	0.01929	0.07931
Salt Lake City	-0.007108	0.03181
Los Angeles	-0.05821	0.007321
San Francisco	-0.02776	0.04787

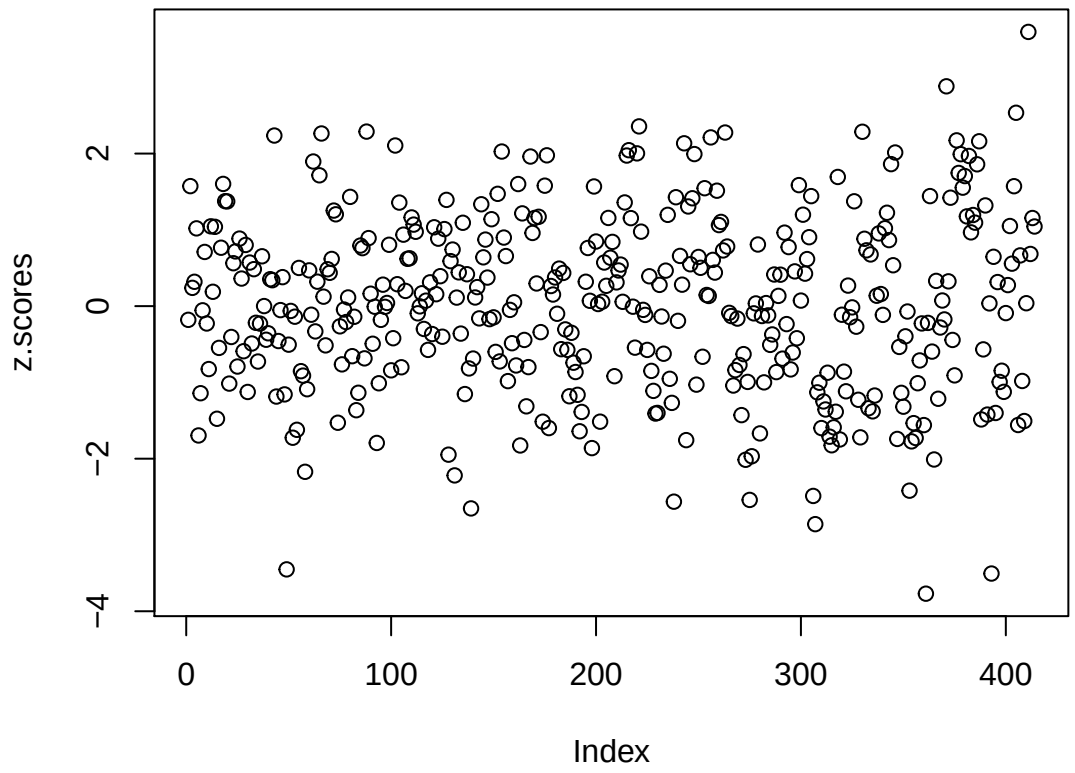
	lower	upper
San Diego	-0.01293	0.04224

```
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")
```

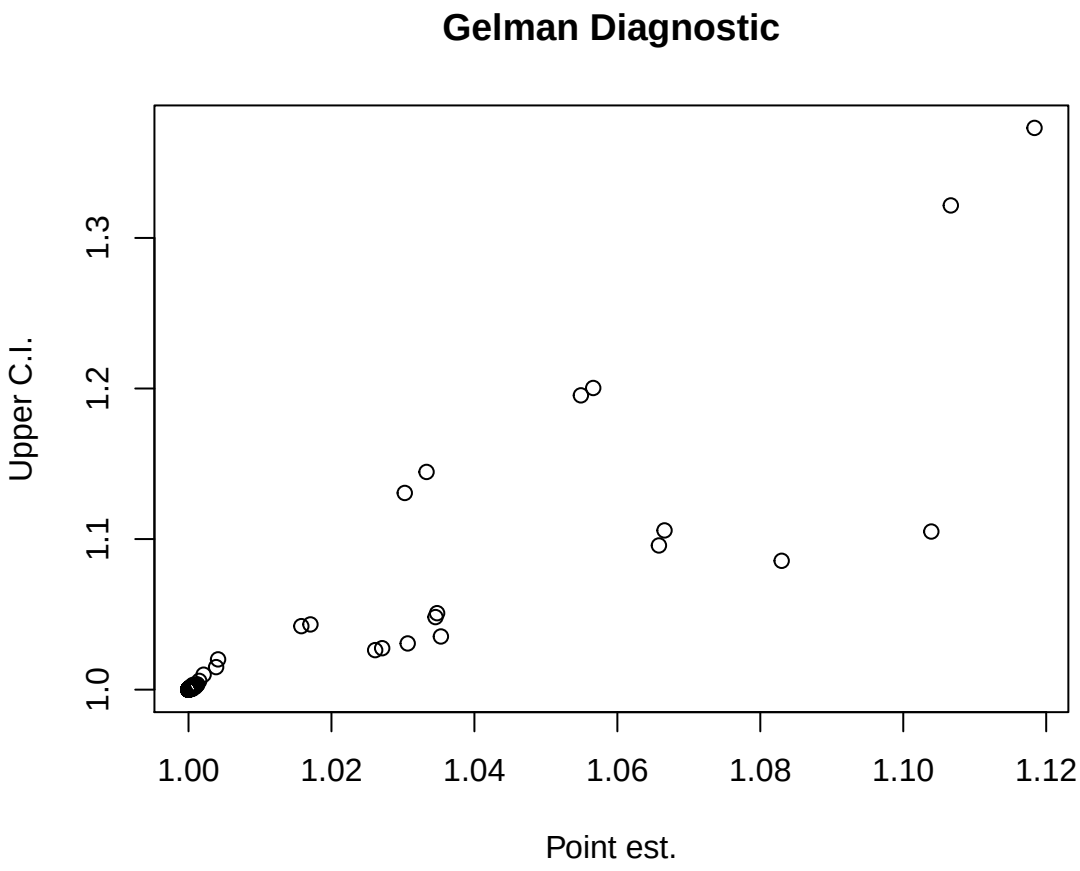
Table 10: Fail count for Heidelberger and Welch diagnotic

fail.count
6

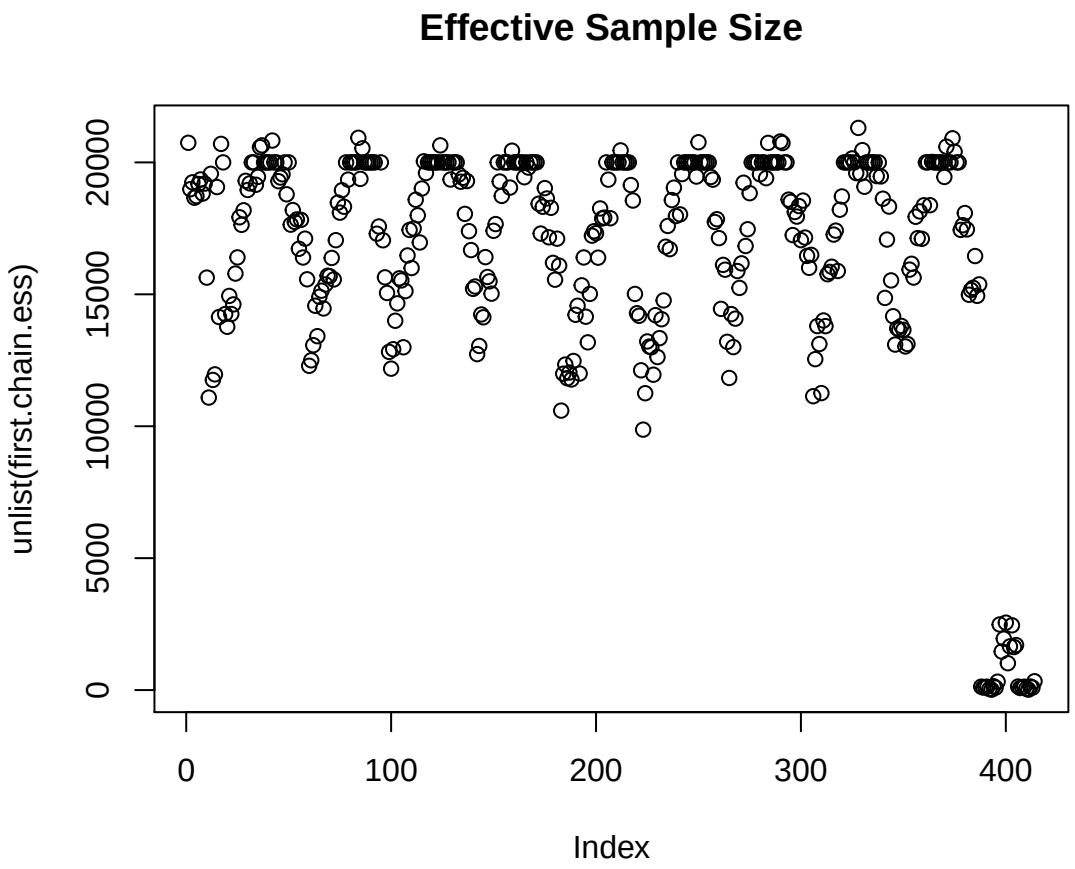
```
# check that our chains length is satisfactory.
#raftery.diag(out.coda) - Indicated 3k so we ran for 40k

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
0.4548	0.2653

pval.mean	pval.sd
0.4628	0.4069
0.4695	0.3821
0.4312	0.2337
0.4768	0.6753
0.4684	0.5951
0.4435	0.0984
0.4677	0.2701
0.47	0.2064

```
####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
1.108	1.184	1.561

DIC Calculation

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

```
dic_pois
```

```
## Mean deviance:  957.9
## penalty 17.56
## Penalized deviance: 975.5
```

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

```
## Mean deviance:  957.7
## penalty 19.61
## Penalized deviance: 977.3
```