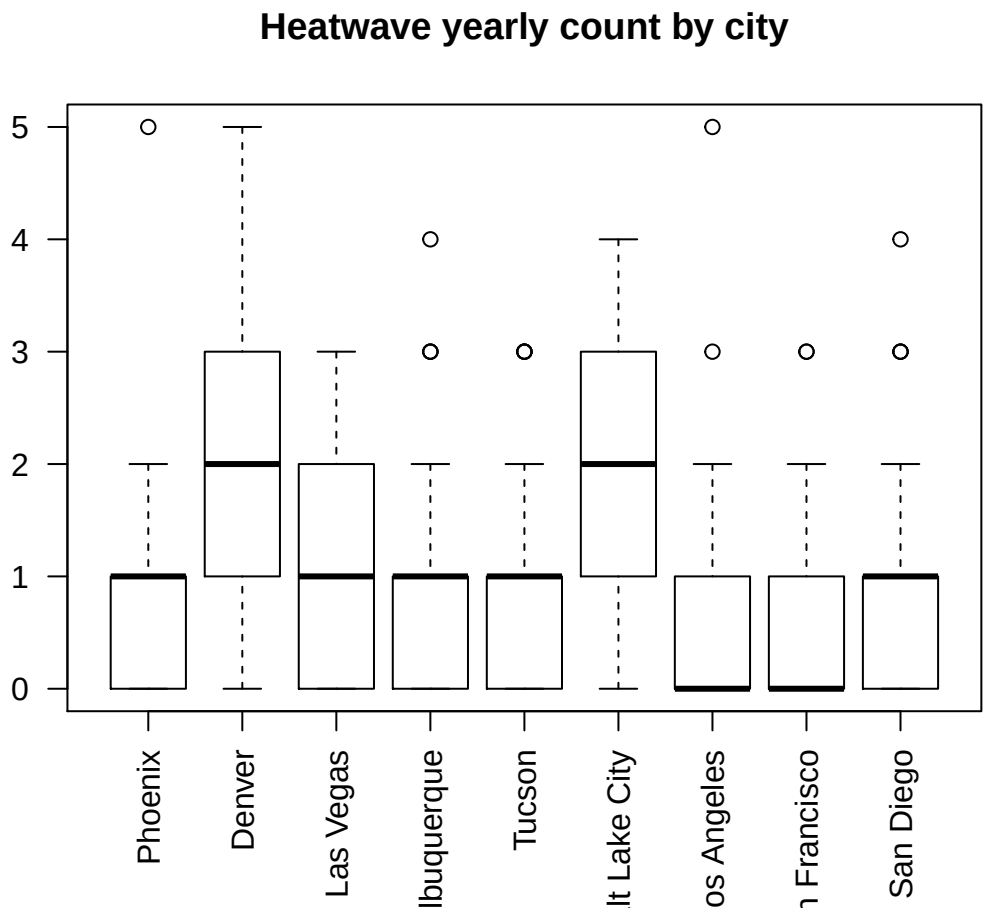
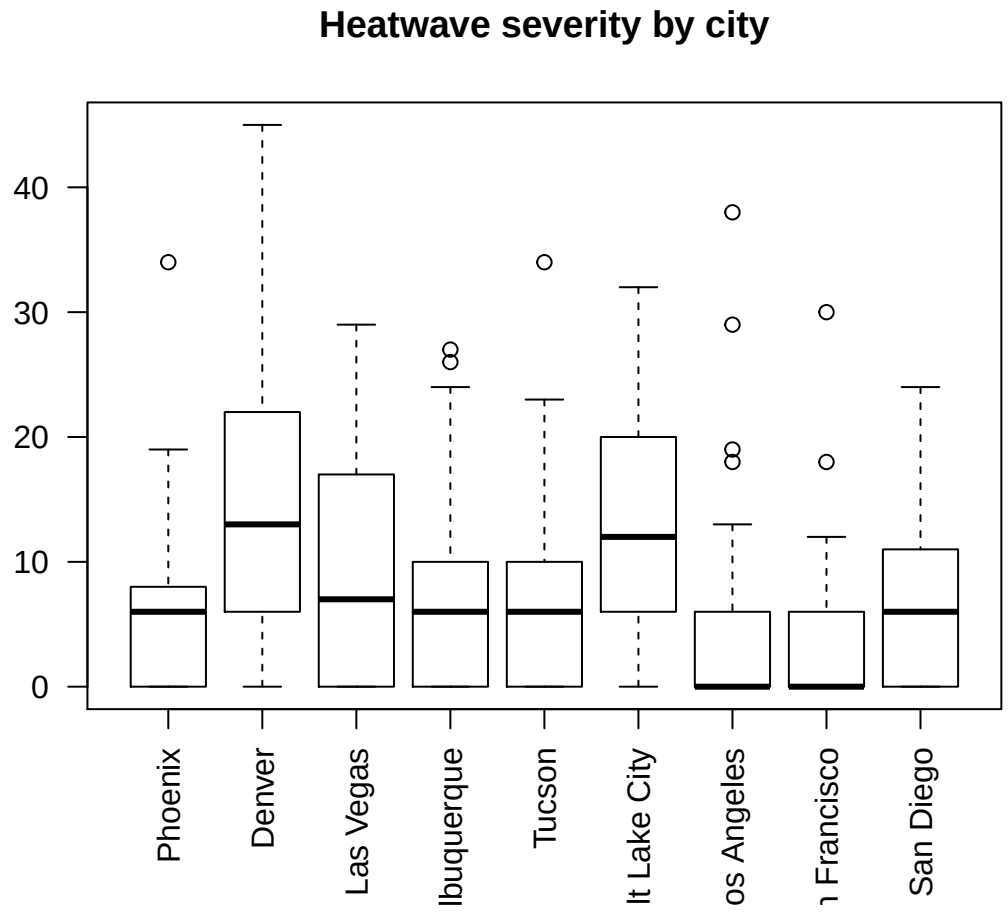
# E3

```
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
nSamples = 5000
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")
```



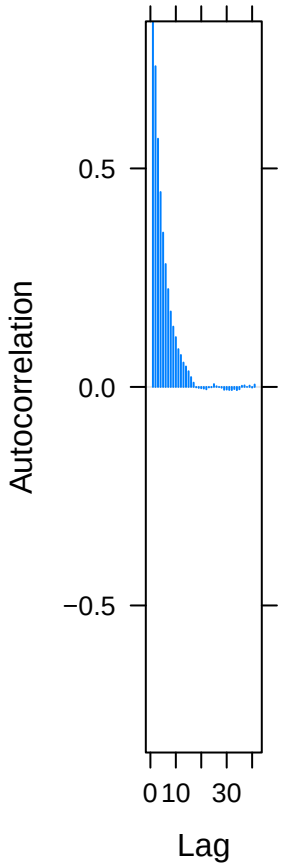

## Fit JAGS Poisson Random Effects

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

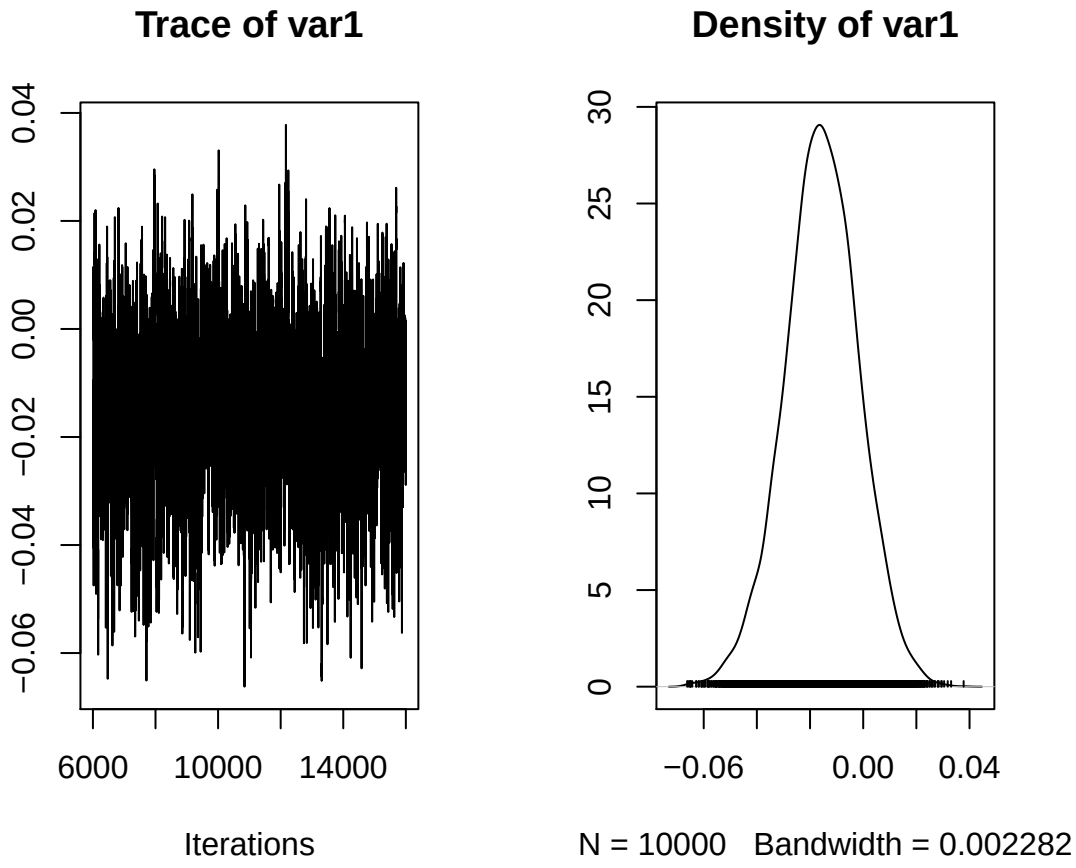
```
  # Slopes
  for(i in 1:9){
    beta[i] ~ dnorm(mu.beta,taus.beta[i])
    taus.beta[i] ~ dgamma(0.1,0.1)
  }

  ## 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)
  #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

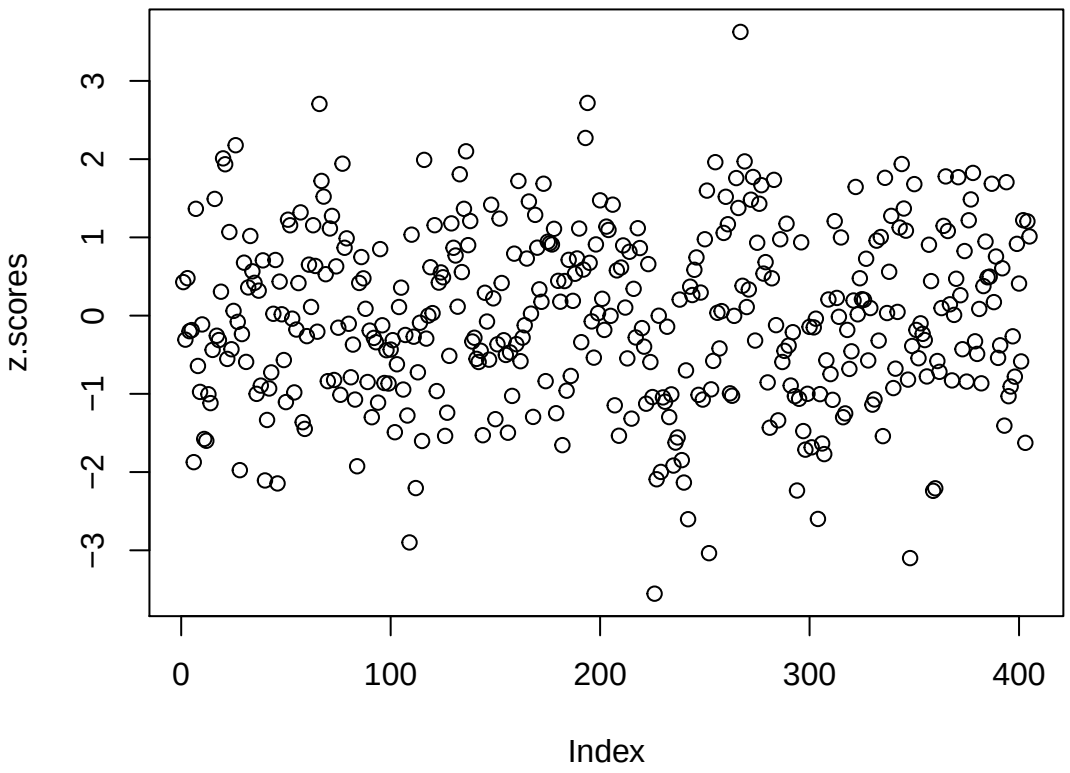
| | lower | upper |
|---|---|---|
| **Phoenix** | -0.04355 | 0.01149 |
| **Denver** | 0.0105 | 0.04461 |
| **Las Vegas** | -0.01407 | 0.03163 |
| **Albuquerque** | -0.04996 | 0.002621 |
| **Tucson** | 0.01796 | 0.07715 |
| **Salt Lake City** | -0.003368 | 0.03383 |
| **Los Angeles** | -0.056 | 0.005243 |
| **San Francisco** | -0.02898 | 0.04481 |
| **San Diego** | -0.01601 | 0.03793 |

```
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")
```

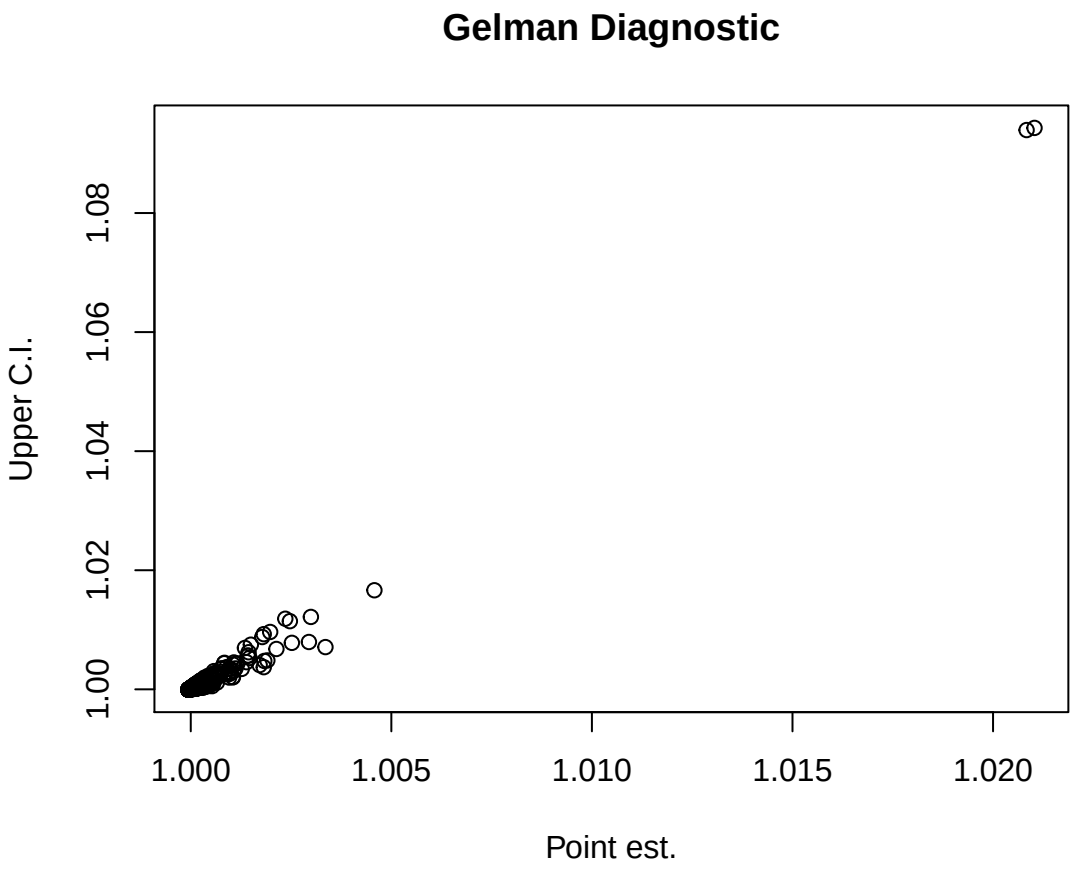
Table 2: Fail count for Heidelberger and Welch diagnotic

| fail.count |
|---|
| 5 |

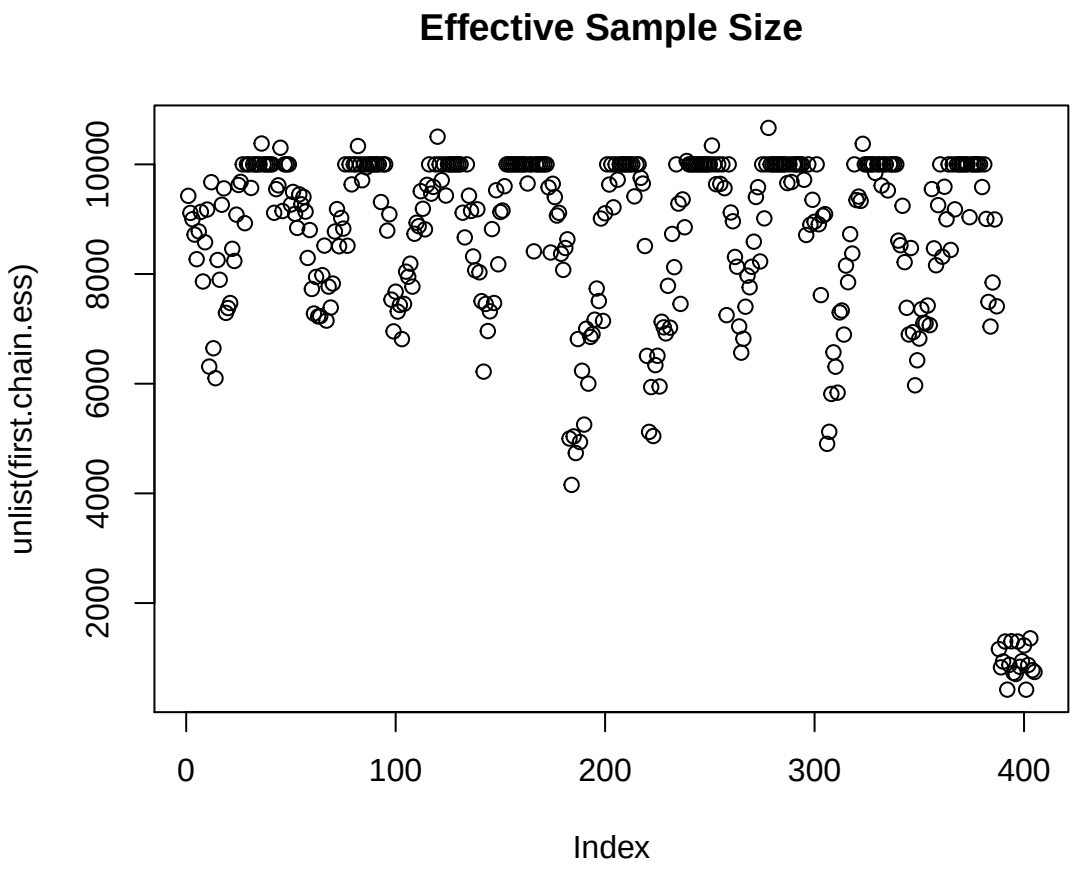
```
# check that our chains length is satisfactory.
#raftery.diag(out.coda) - Indicated 3k so we ran for 40k

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 3: Baeysian p-values Poisson GLM

| pval.mean | pval.sd |
|---|---|
| 0.4509 | 0.2646 |

| pval.mean | pval.sd |
|---|---|
| 0.4706 | 0.4175 |
| 0.474 | 0.3879 |
| 0.4137 | 0.2165 |
| 0.4866 | 0.6731 |
| 0.4343 | 0.5917 |
| 0.4394 | 0.0956 |
| 0.4982 | 0.2881 |
| 0.4936 | 0.209 |

```
####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 |
|---|---|---|
| 1.107 | 1.176 | 1.591 |

## Fit JAGS Negative Binomial Random Effects

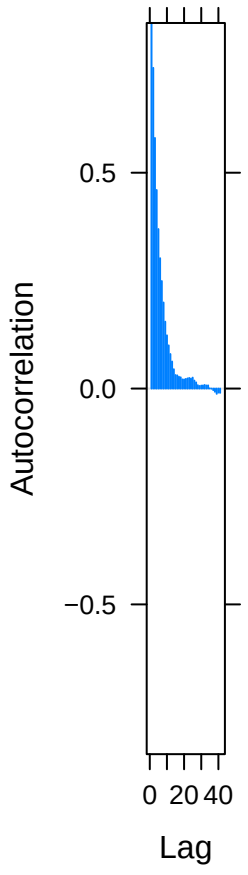
```
######################################## Fit JAGS Negative Binomial Random Effects
model_nb = '
```

```
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)
  #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

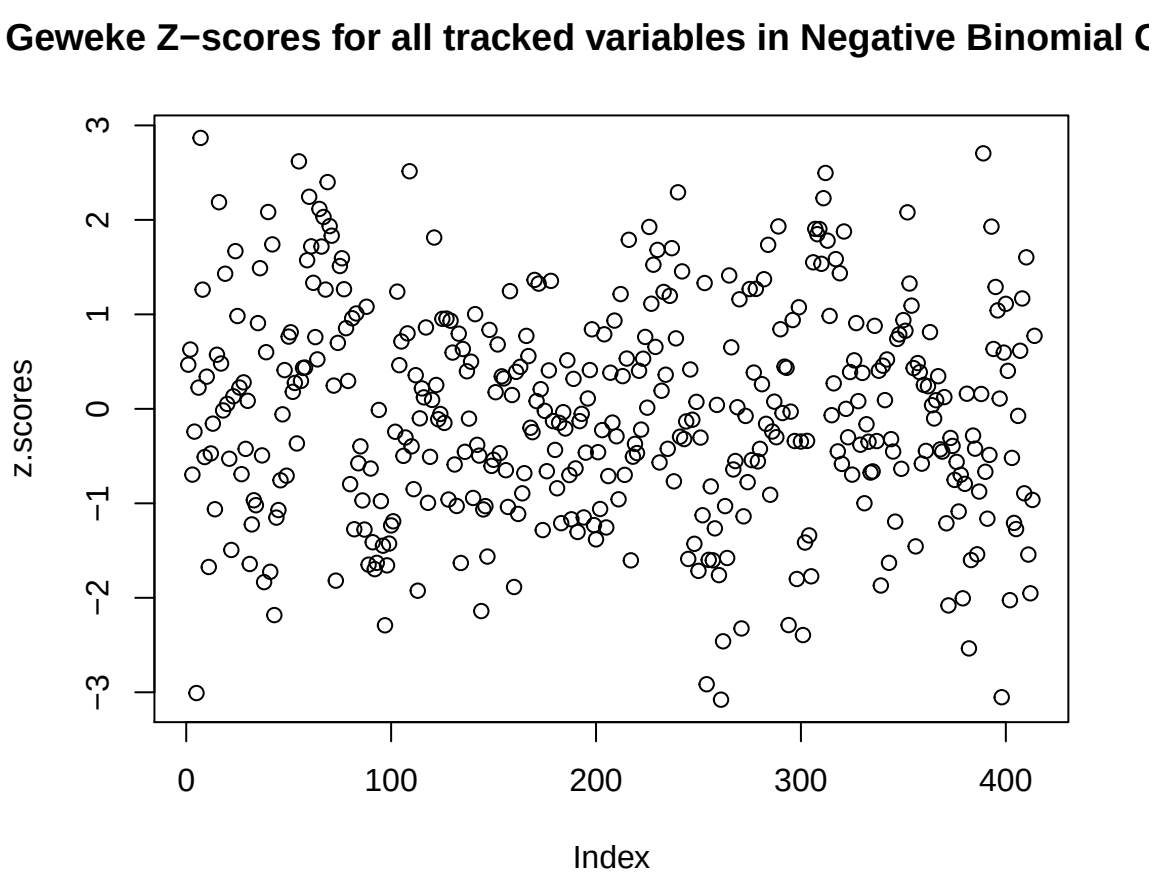
| | lower | upper |
|---|---|---|
| **Phoenix** | -0.04721 | 0.01387 |
| **Denver** | 0.007373 | 0.04819 |
| **Las Vegas** | -0.01465 | 0.0343 |
| **Albuquerque** | -0.05591 | 0.005594 |
| **Tucson** | 0.01427 | 0.08104 |
| **Salt Lake City** | -0.006478 | 0.03495 |
| **Los Angeles** | -0.06038 | 0.006179 |
| **San Francisco** | -0.03193 | 0.04858 |
| **San Diego** | -0.01495 | 0.04215 |

```
#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")
```

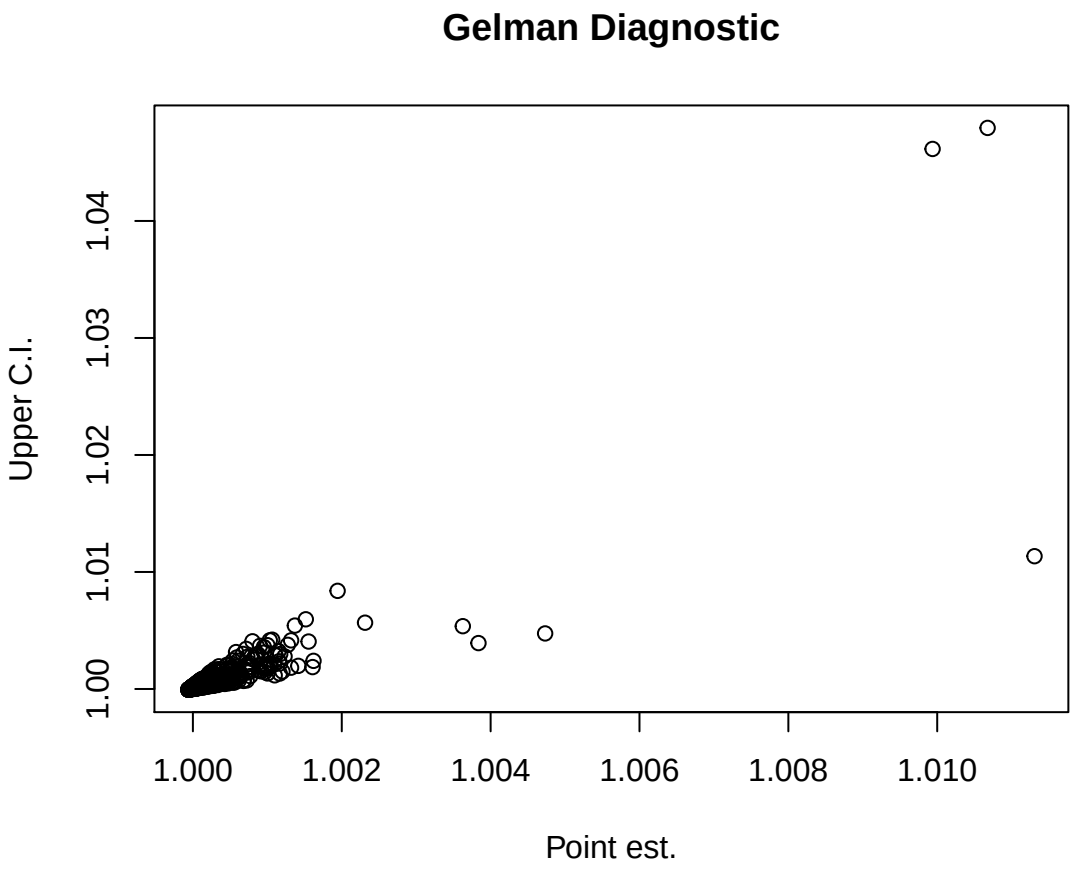
Table 6: Fail count for Heidelberger and Welch diagnotic

| fail.count |
|---|
| 3 |

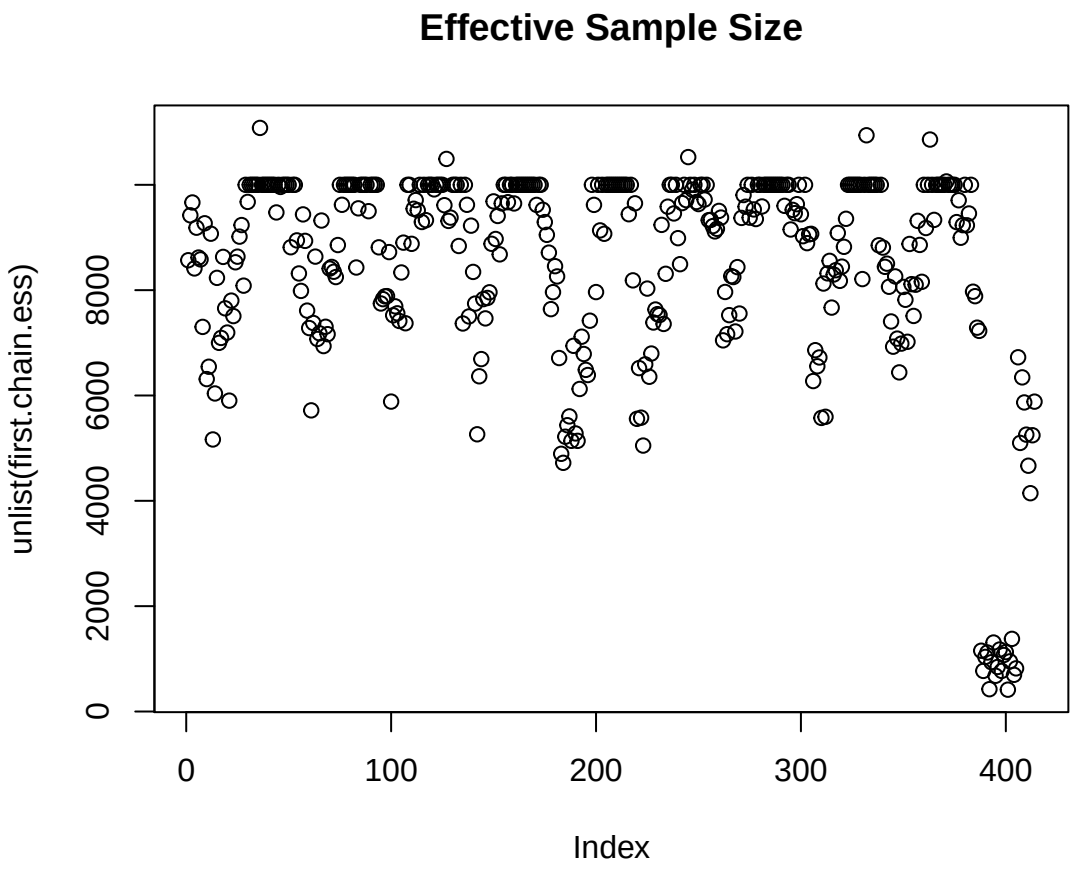
```
# check that our chains length is satisfactory.
#raftery.diag(out.coda) - Indicated 3k so we ran for 40k

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 |
|---|---|
| 0.462 | 0.431 |

| pval.mean | pval.sd |
|---|---|
| 0.4813 | 0.6883 |
| 0.4946 | 0.607 |
| 0.4381 | 0.4288 |
| 0.5083 | 0.7537 |
| 0.4564 | 0.7997 |
| 0.4587 | 0.2756 |
| 0.509 | 0.4313 |
| 0.5152 | 0.4416 |

```
####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 |
|---|---|---|
| 1.108 | 1.176 | 1.748 |

## DIC Calculation

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

```
dic_pois
```

```
## Mean deviance:  957.4
## penalty 16.53
## Penalized deviance: 973.9
```

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

```
## Mean deviance:  957.5
## penalty 19.22
## Penalized deviance: 976.7
```