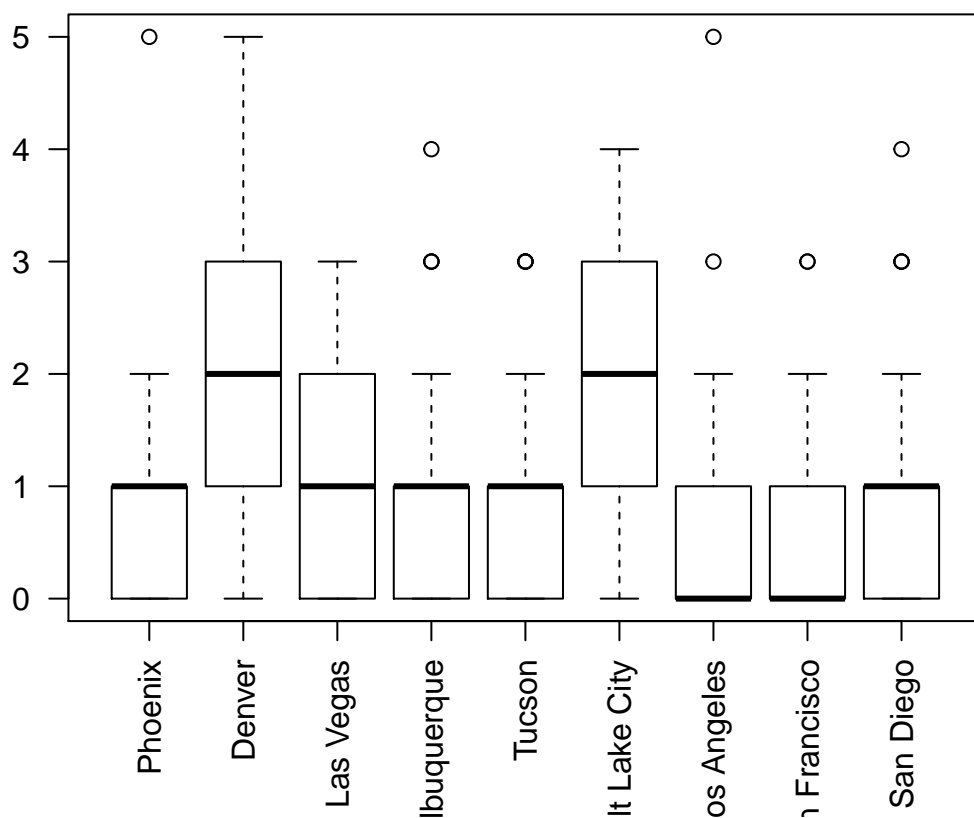


## E3

```
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 = 1000
load("HWD2.RData")

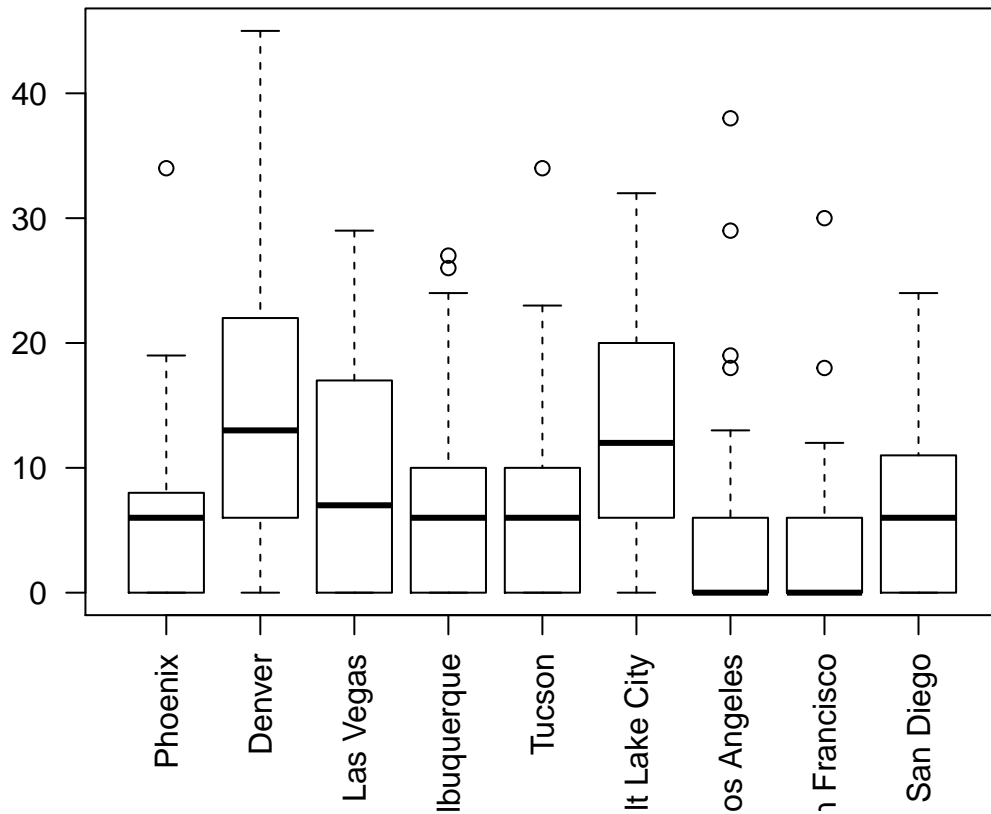
df <- data.frame(X.num)
colnames(df) <- city_names
boxplot(df, las = 2, main = "Heatwave yearly count by city")
```

**Heatwave yearly count by city**



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

## 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]
  }
}
}
'

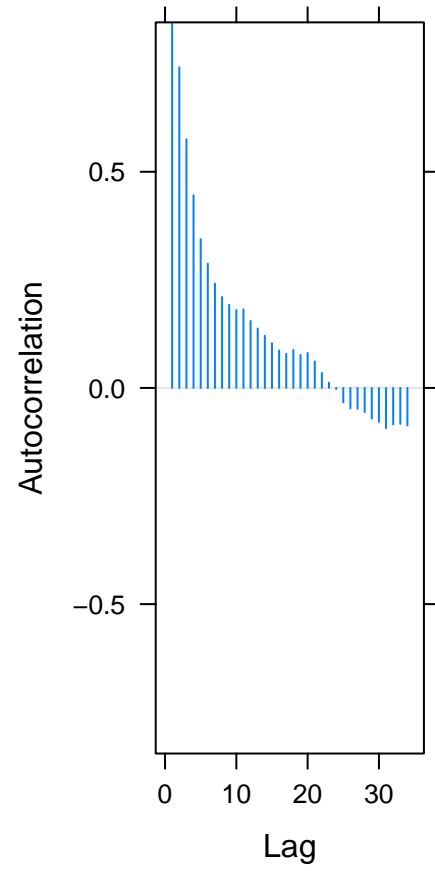
# Set up the data
model_data = list(N = 41, t=seq(1:41),Y=X.num,mu.beta=0,tau.beta=.0001,mu.intercept=0,tau.intercept=.
# Choose the parameters to watch
model_parameters = c("beta", "alpha","Dm", "Dsd", "Yp")
model_pois <- jags.model(textConnection(model_pois),data = model_data,n.chains = n.chains)#Compile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

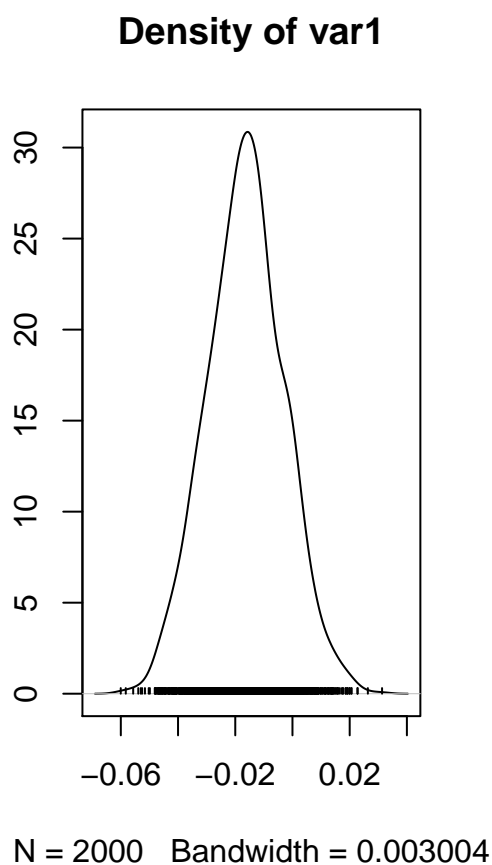
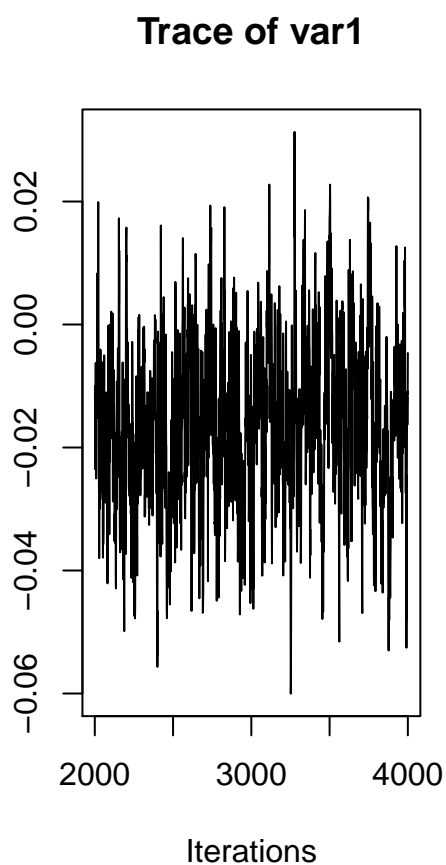
update(model_pois, nSamples, progress.bar="none"); # Burnin
out.coda <- coda.samples(model_pois, variable.names=model_parameters,n.iter=2*nSamples,n.thin=2)
#plot(out.coda)

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	-0.04311	0.009168
Denver	0.01165	0.04548
Las Vegas	-0.01388	0.03243
Albuquerque	-0.04599	0.005101
Tucson	0.01737	0.07403
Salt Lake City	-0.00374	0.03326
Los Angeles	-0.05475	0.004613
San Francisco	-0.02916	0.04165
San Diego	-0.0179	0.03498

```

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 Heidelberg and Welch diagnostic")

```

Table 2: Fail count for Heidelberg and Welch diagnostic

fail.count
2

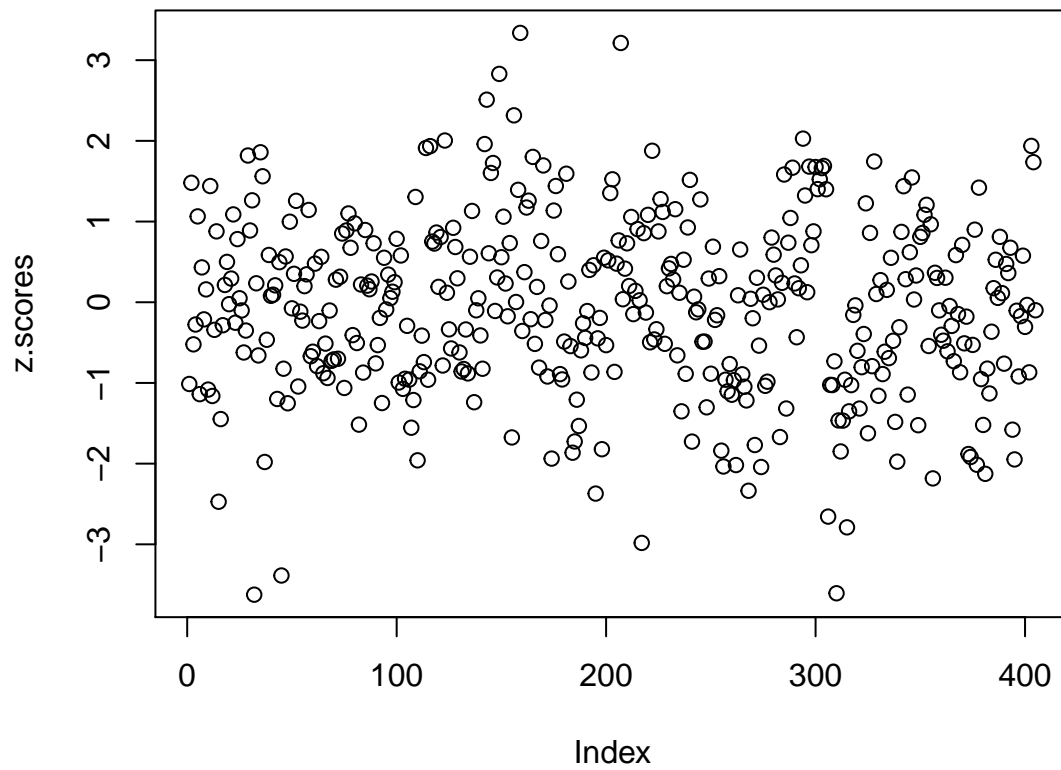
```

# 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")

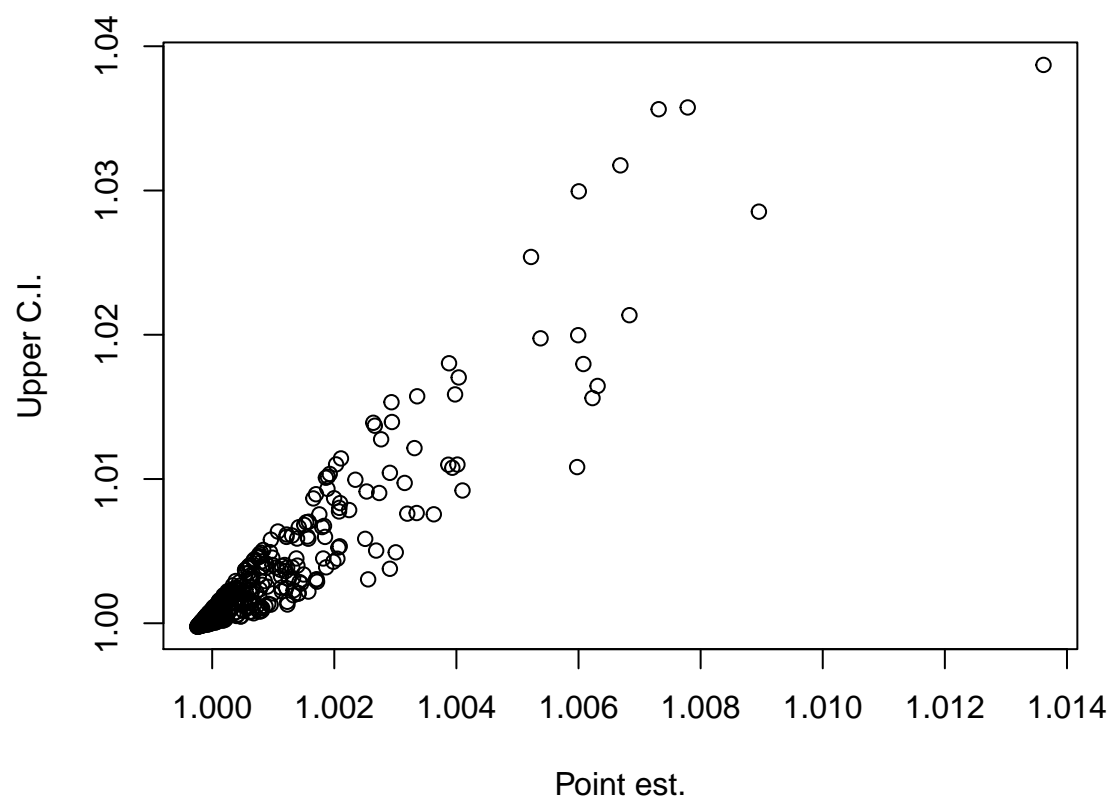
```

## Geweke Z-scores for all tracked variables in Poisson GLM



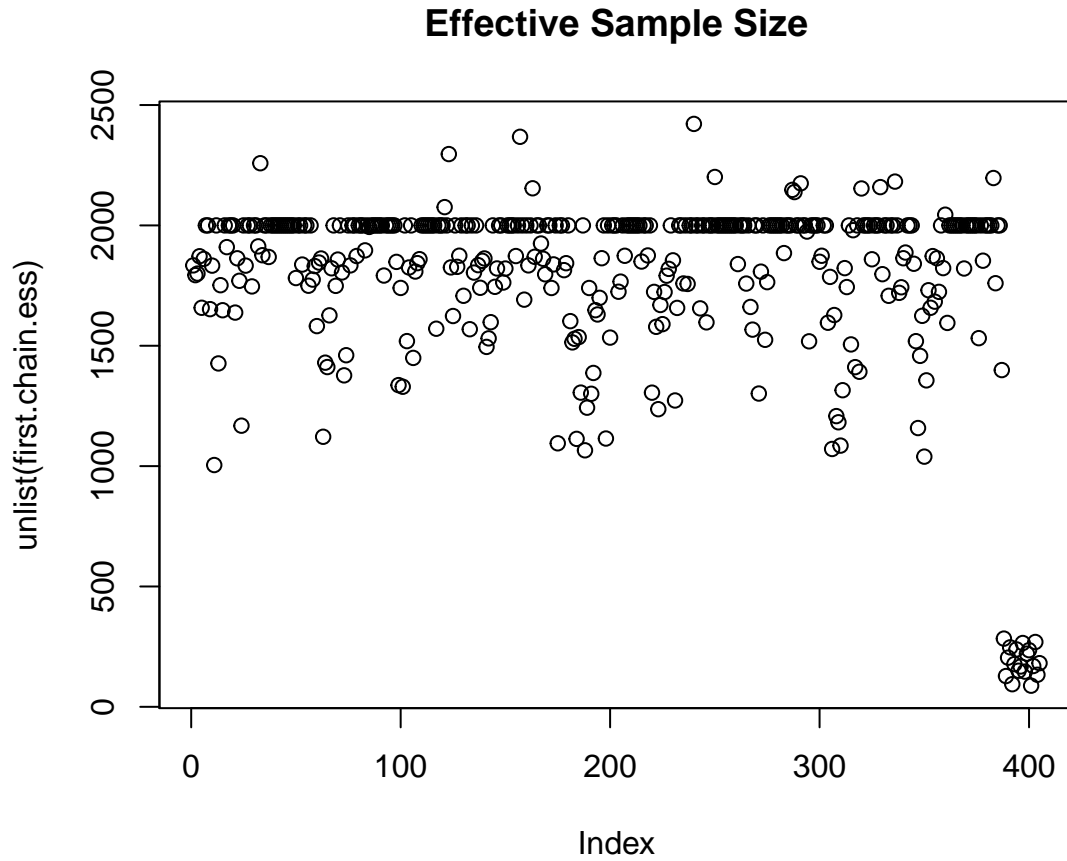
```
if(n.chains > 1)
{
  gelman.srf <-gelman.diag(out.coda)
  plot(gelman.srf$psrf,main = "Gelman Diagnostic")
}
```

## 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 = "Baeyesian p-values Poisson GLM")
```

Table 3: Baeyesian p-values Poisson GLM

pval.mean	pval.sd
0.4635	0.274

pval.mean	pval.sd
0.4495	0.4145
0.466	0.382
0.408	0.2075
0.47	0.6375
0.429	0.594
0.4195	0.0825
0.5185	0.2995
0.498	0.197

```
####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(idi,"")[[1]][1])
    idj <- as.numeric(strsplit(idj,"")[[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(idi,"")
    idi <- as.numeric(strsplit(idi,"")[[1]][1])
    idj <- as.numeric(strsplit(idj,"")[[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=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
1.112	1.174	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=.0)
# 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

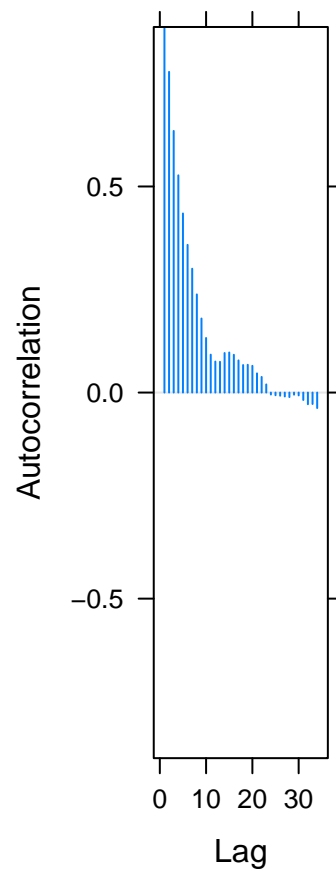
```

```

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
<b>Phoenix</b>	-0.05078	0.01604
<b>Denver</b>	0.008249	0.04727
<b>Las Vegas</b>	-0.0165	0.03631
<b>Albuquerque</b>	-0.0582	0.006812
<b>Tucson</b>	0.01575	0.07754
<b>Salt Lake City</b>	-0.005004	0.03807
<b>Los Angeles</b>	-0.06089	0.01113
<b>San Francisco</b>	-0.02897	0.04525
<b>San Diego</b>	-0.01555	0.03759

```

#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 diagnostic")

```

Table 6: Fail count for Heidelberger and Welch diagnostic

fail.count
8

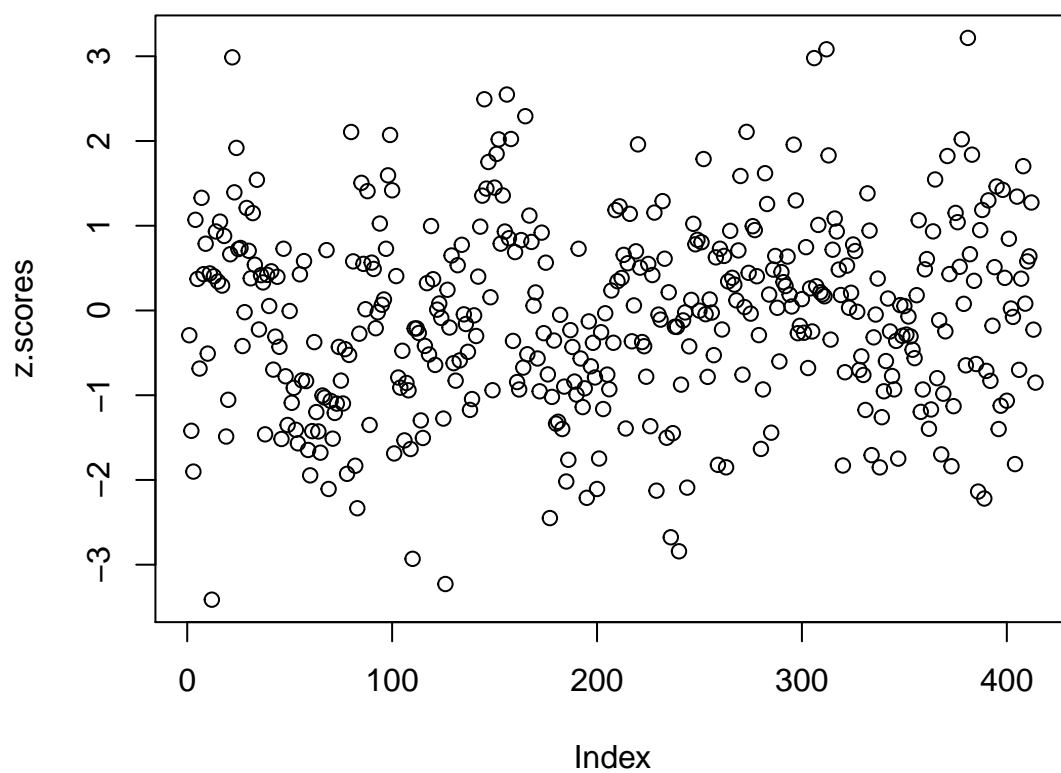
```

# 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")

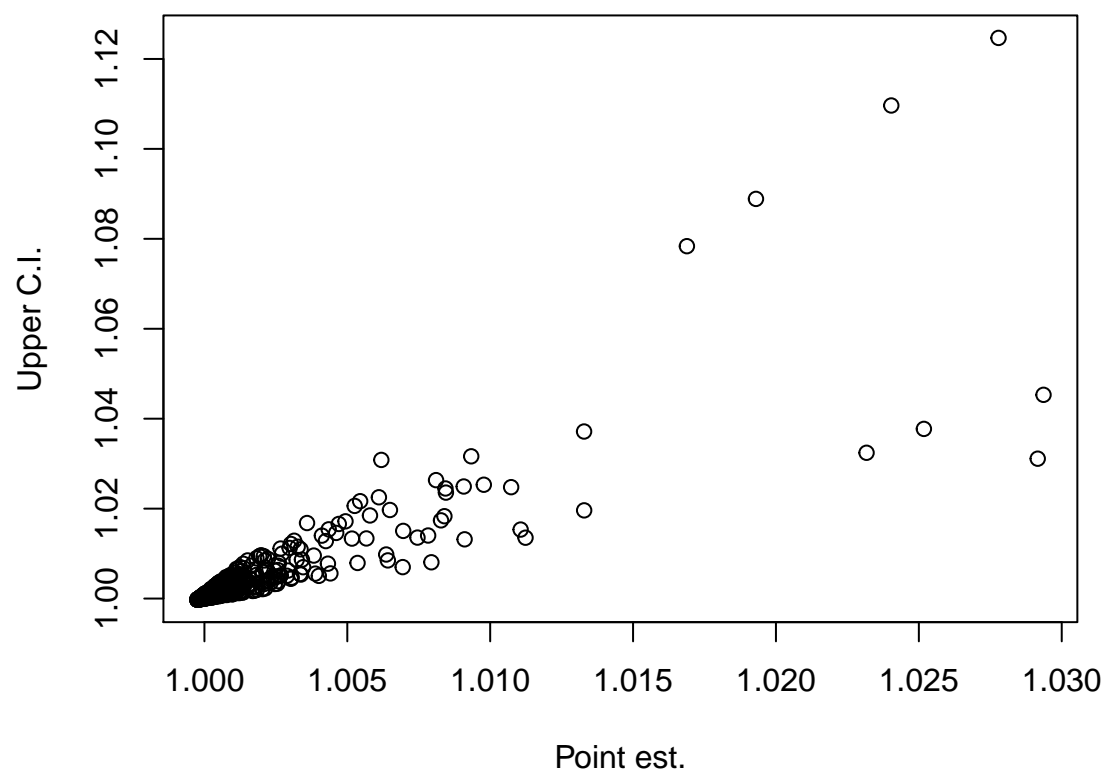
```

## Geweke Z-scores for all tracked variables in Negative Binomial C



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

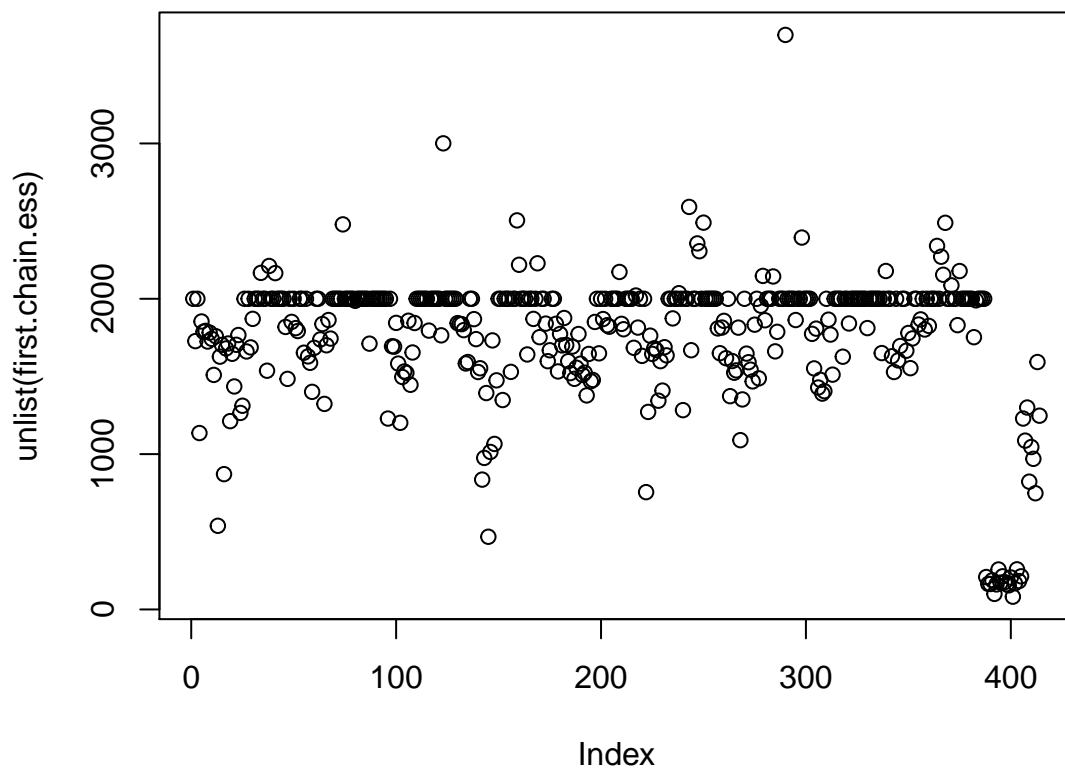
## Gelman Diagnostic



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



## 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 = "Baeyesian p-values Poisson GLM")
```

Table 7: Baeyesian p-values Poisson GLM

pval.mean	pval.sd
0.456	0.435

pval.mean	pval.sd
0.486	0.6835
0.4775	0.6045
0.456	0.4575
0.4975	0.761
0.4725	0.8125
0.473	0.294
0.5165	0.4225
0.516	0.435

```
####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(idi,"")[1][1])
    idj <- as.numeric(strsplit(idj,"")[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(idi,"")
    idi <- as.numeric(strsplit(idi,"")[1][1])
    idj <- as.numeric(strsplit(idj,"")[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=train.mse.mode)
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
1.115	1.174	1.77

## Fit JAGS Poisson GLM With Latitude

```

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

```

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

```

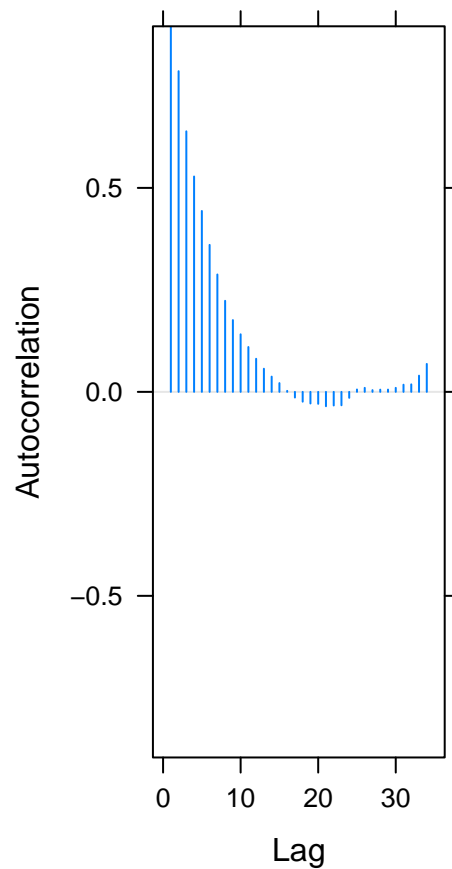
# Set up the data
model_data = list(N = 41, t=seq(1:41),Y=X.num,lattitude=lattitude )
# Choose the parameters to watch
model_parameters = c("beta", "alpha","gamma","Dm", "Dsd", "Yp")
model_pois <- jags.model(textConnection(model_pois),data = model_data,n.chains = n.chains)#Compile Mo

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

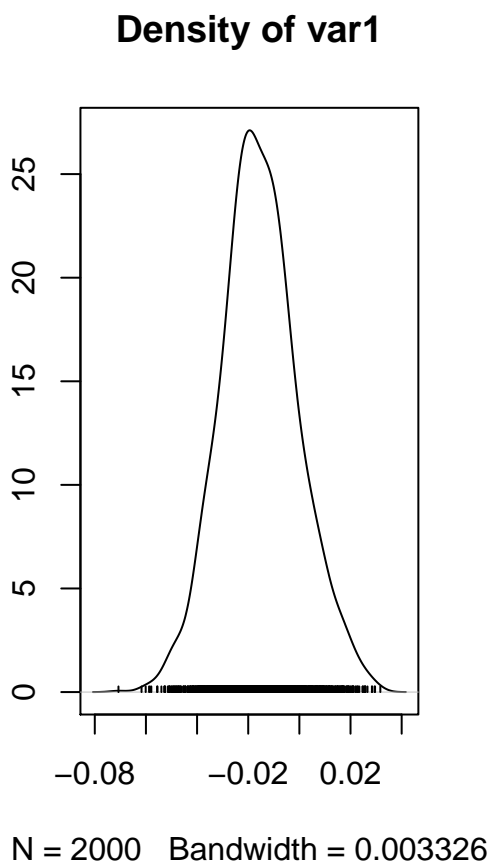
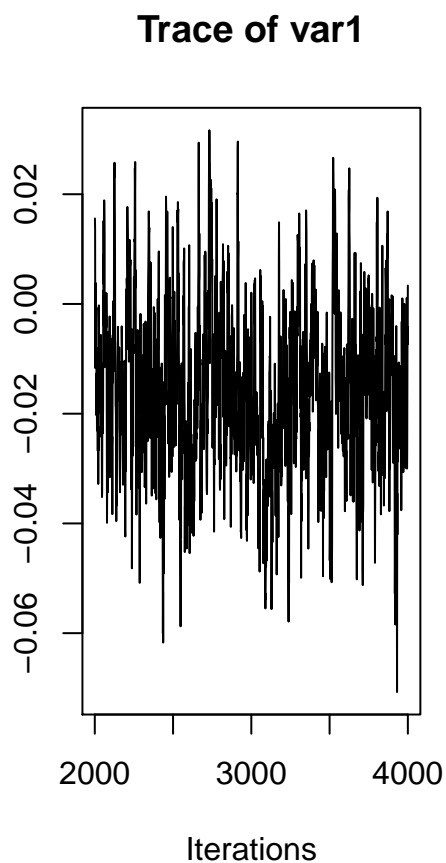
update(model_pois, nSamples, progress.bar="none"); # Burnin
out.coda <- coda.samples(model_pois, variable.names=model_parameters,n.iter=2*nSamples,,n.thin=2)
#plot(out.coda)

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	-0.04269	0.01718
Denver	0.009941	0.04739
Las Vegas	-0.01139	0.03539
Albuquerque	-0.05007	0.002473
Tucson	0.01912	0.08022
Salt Lake City	-0.00697	0.03079
Los Angeles	-0.05802	0.002158
San Francisco	-0.02479	0.04611

	lower	upper
San Diego	-0.0126	0.04195

```
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 diagnostic")
```

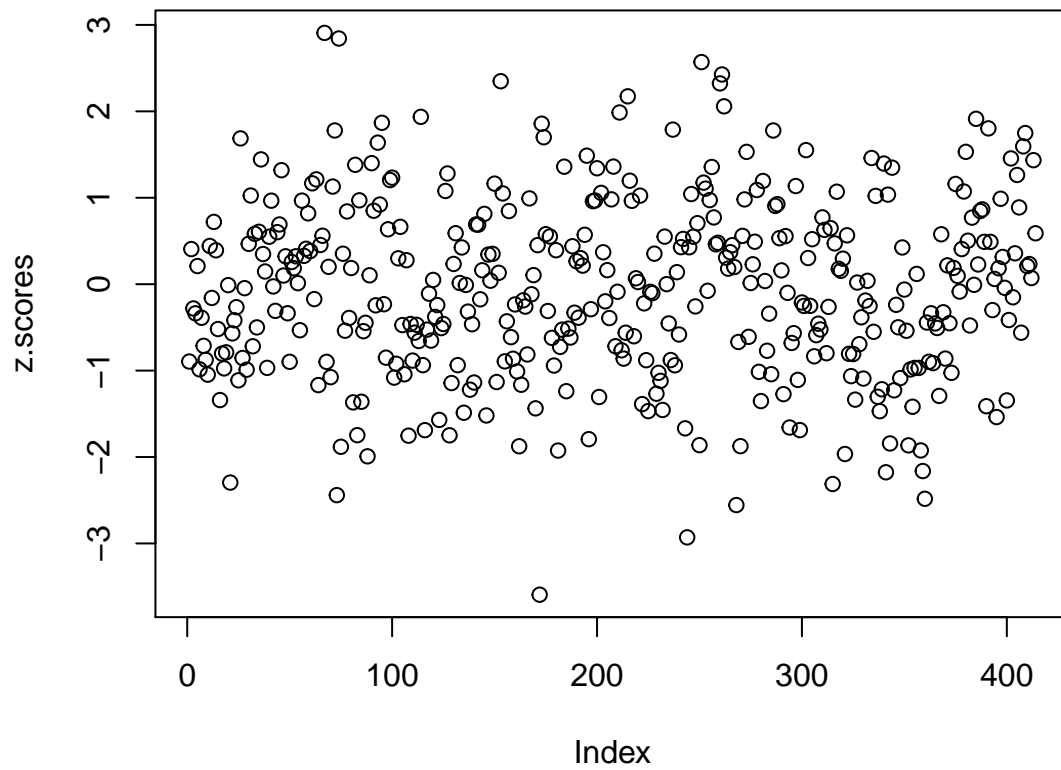
Table 10: Fail count for Heidelberger and Welch diagnostic

fail.count
2

```
# 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")
```

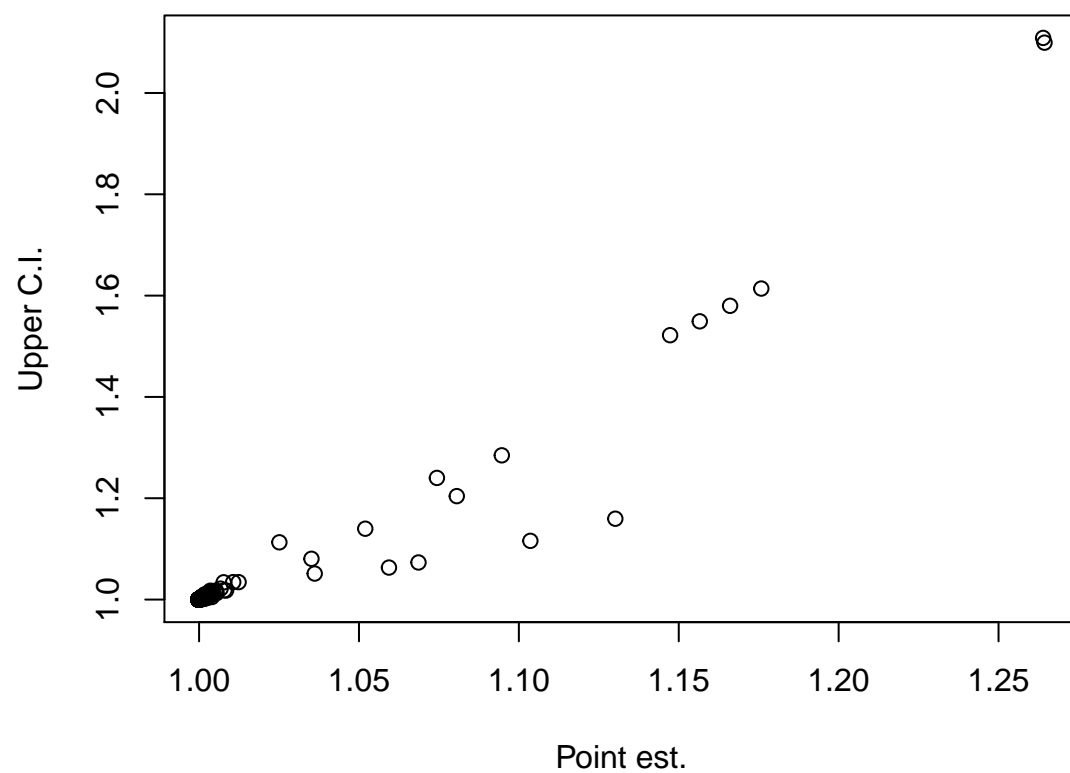
## Geweke Z-scores for all tracked variables in Poisson GLM



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

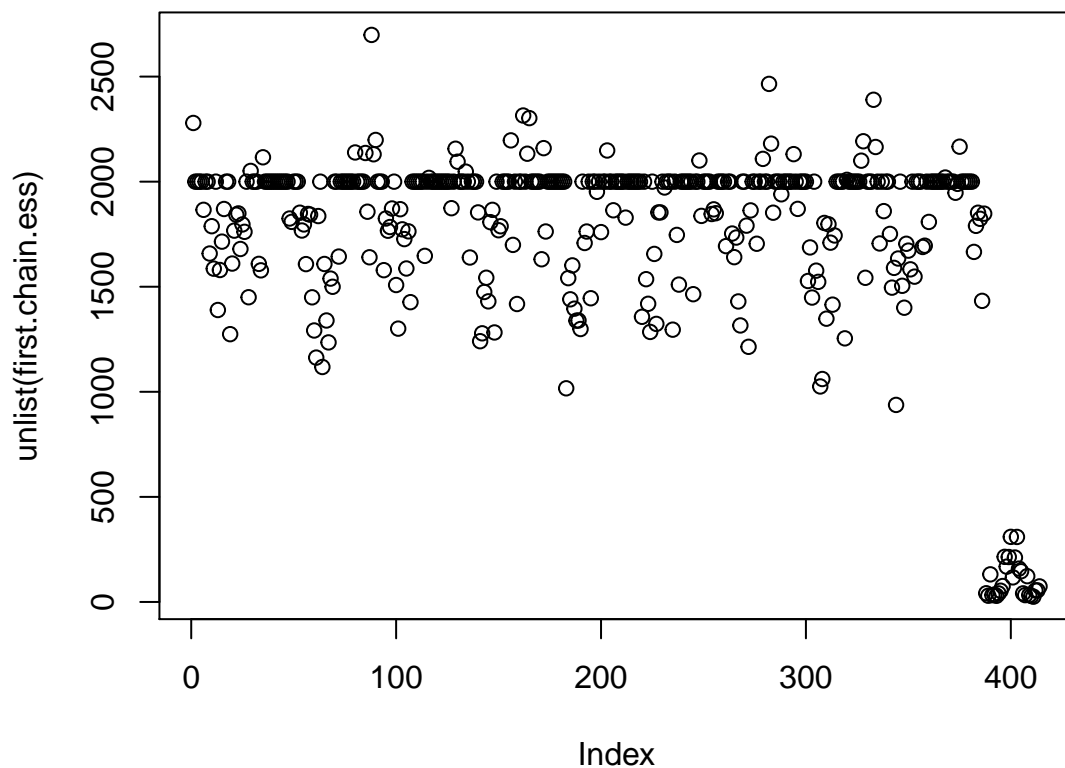


## Gelman Diagnostic



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

## 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 = "Baeyesian p-values Poisson GLM")
```

Table 11: Baeyesian p-values Poisson GLM

pval.mean	pval.sd
0.4275	0.2515

pval.mean	pval.sd
0.47	0.4355
0.479	0.386
0.429	0.234
0.465	0.6515
0.448	0.5855
0.4405	0.095
0.498	0.2685
0.4525	0.203

```
####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(idi,"")[[1]][1])
    idj <- as.numeric(strsplit(idj,"")[[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(idi,"")
    idi <- as.numeric(strsplit(idi,"")[[1]][1])
    idj <- as.numeric(strsplit(idj,"")[[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=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
1.107	1.168	1.534

## DIC Calculation

```

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

```

```
dic_pois
```

```
## Mean deviance: 957.6
```

```
## penalty 17.51
```

```
## Penalized deviance: 975.1
```

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

```
## Mean deviance: 957.7
```

```
## penalty 19.93
```

```
## Penalized deviance: 977.6
```