Bayesian Modelling - Assessed Homework

Student Name: Harleen Gulati , Student Number: 2101550

2024-04-01

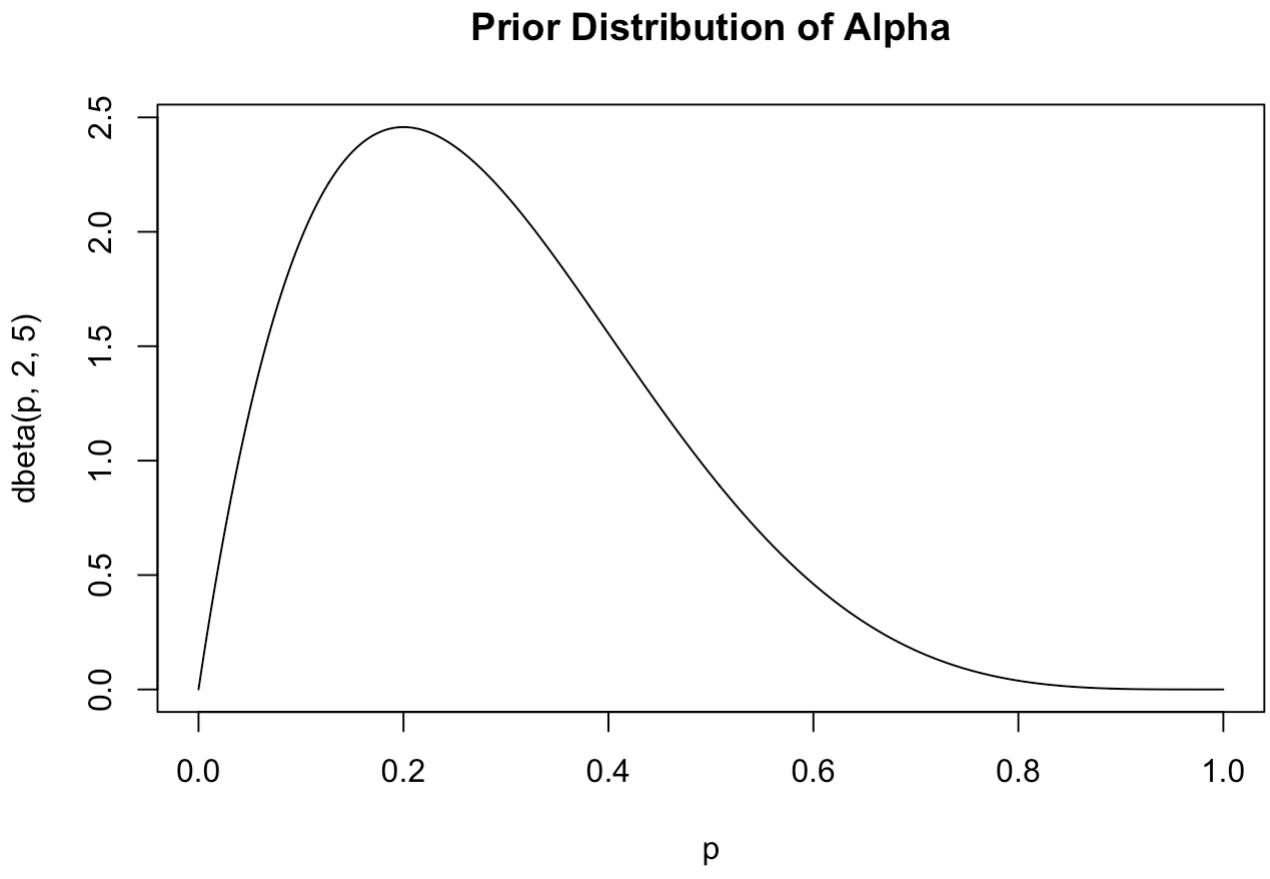
Question 1.

α has an informative prior.

The plot below suggests the prior believes α to lie between 0.1 and 0.3 and strongly believes α to lie at around 0.2.

The prior believes α to be unlikely greater than 0.4 and less than 0.1.

```
p = seq(0, 1, length=1000)
plot(p, dbeta(p, 2, 5), type ='l', main='Prior Distribution of Alpha')
```



Question 2.

Begin by defininig the model.

```
modelstring = "model {
  alpha ~ dbeta(2, 5)
  mu ~ dexp(0.01)
  tau ~ dexp(10)
  sd = pow(tau, -0.5)
  for (i in 2:N) {
    x[i] ~ dnorm(mu + alpha*(x[i-1]-mu) , tau)
  }
}"
```

Get the data and initialise the model.

```
lake_huron = LakeHuron # gets the data

jags_data = list(x=lake_huron, N=98)

# initialise the model with 4 MCMC chains
model = jags.model(
    textConnection(modelstring),
    data=jags_data,
    n.chains=4
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 97
##    Unobserved stochastic nodes: 3
##    Total graph size: 367
##
## Initializing model
```

Run burn-in for 1000 observations and then perform MCMC for 1000 time steps.

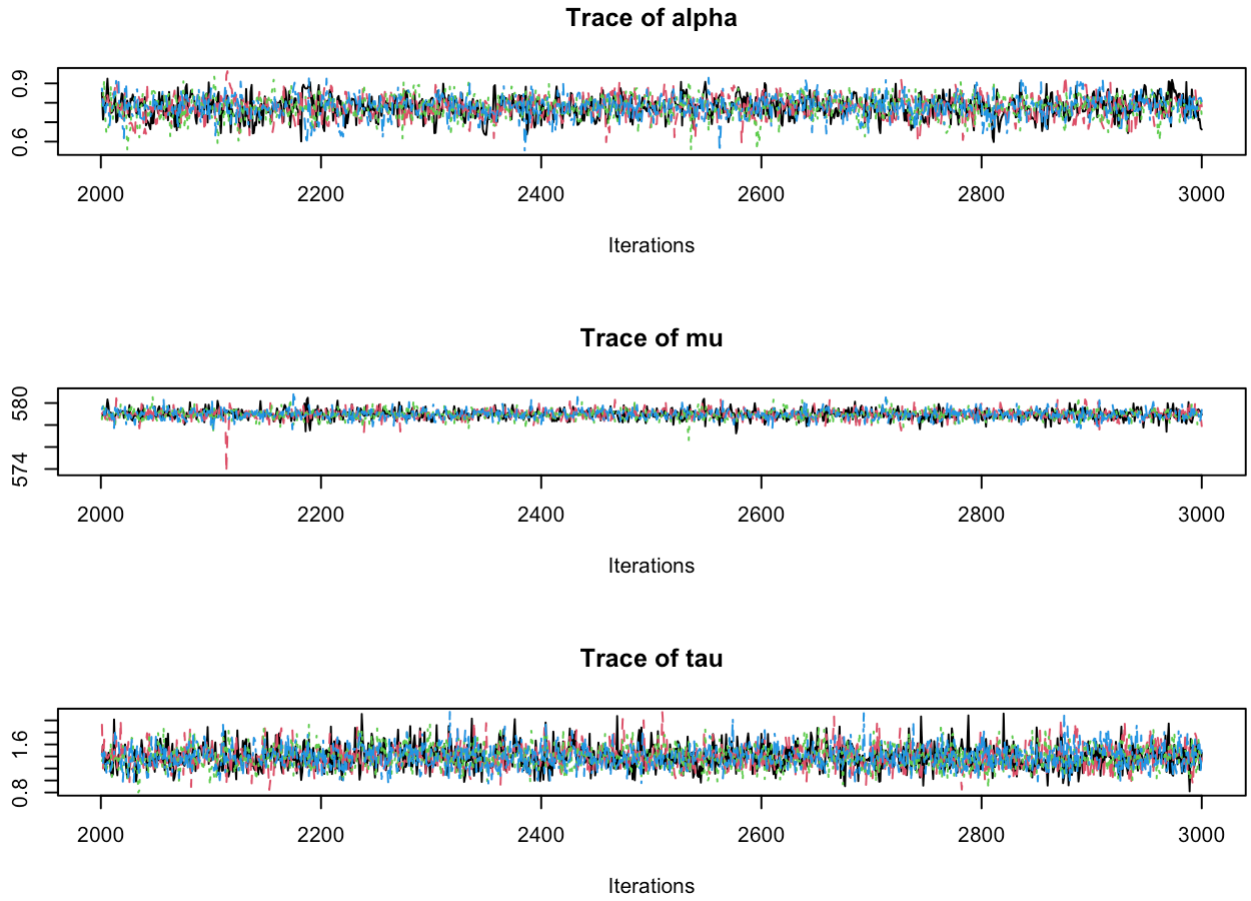
```
# burn-in
update(model, n.iter=1000)

# MCMC for 1000 time steps
samples = coda.samples(model = model, variable.names=c("alpha", "mu", "tau"), n.iter=
1000)
```

We observe the trace plots for all the chains shown below.

The trace plot for alpha suggests it could do with better mixing.

```
par(mfrow=c(3,1))
traceplot(samples)
```

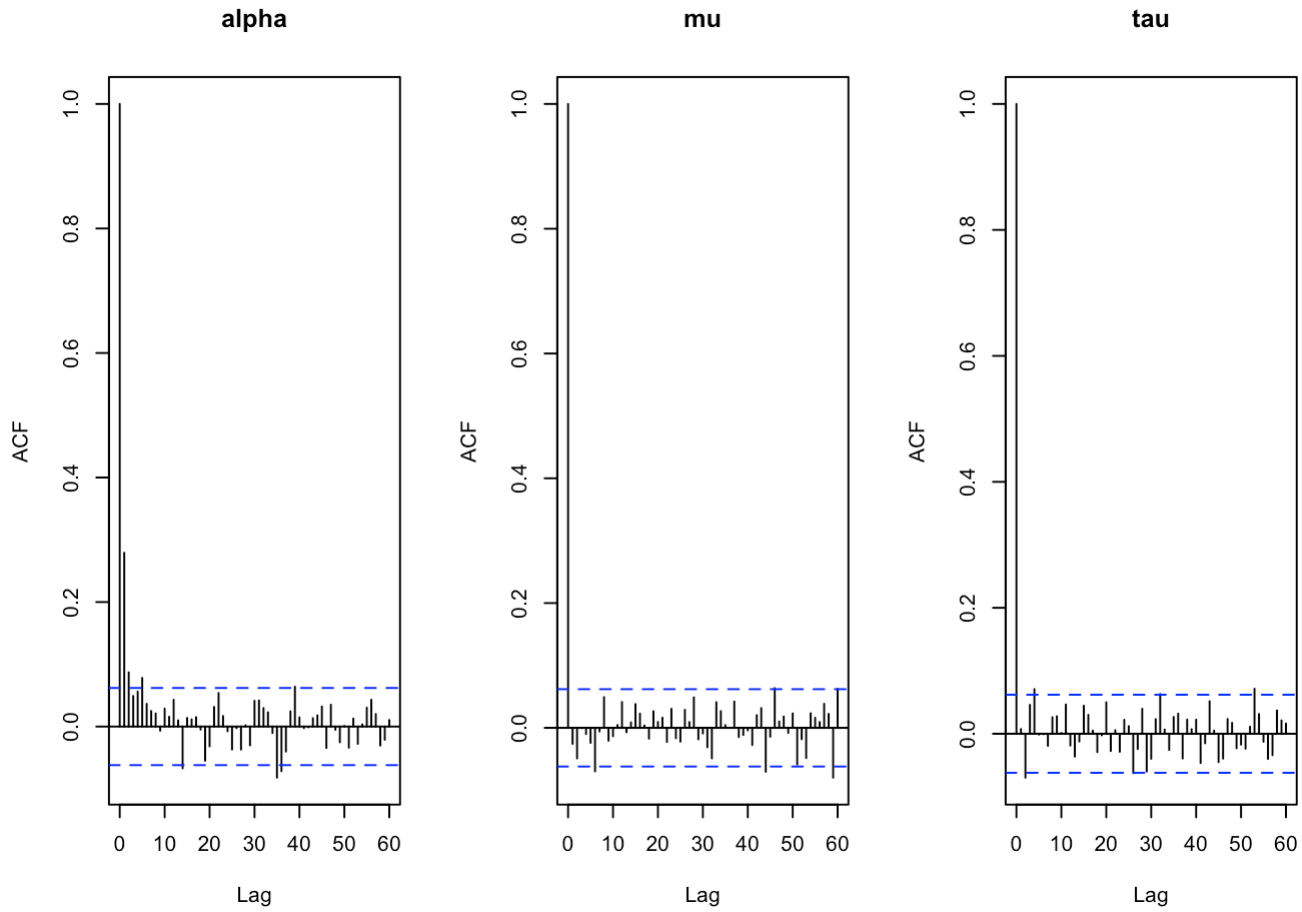


We now observe autocorrelation plots for chain 1. The autocorrelation plots suggest some dependence and so we observe when the ACF dips below the significance level for each parameter.

For α this happens at around time 3, for μ at around time 1 and for τ at around time 2 - we thin to the largest of the three values, adding a bit extra to ensure no autocorrleation remains (i.e., chain 1 suggests a thinning of 9 - the little bit extra we add here is 6).

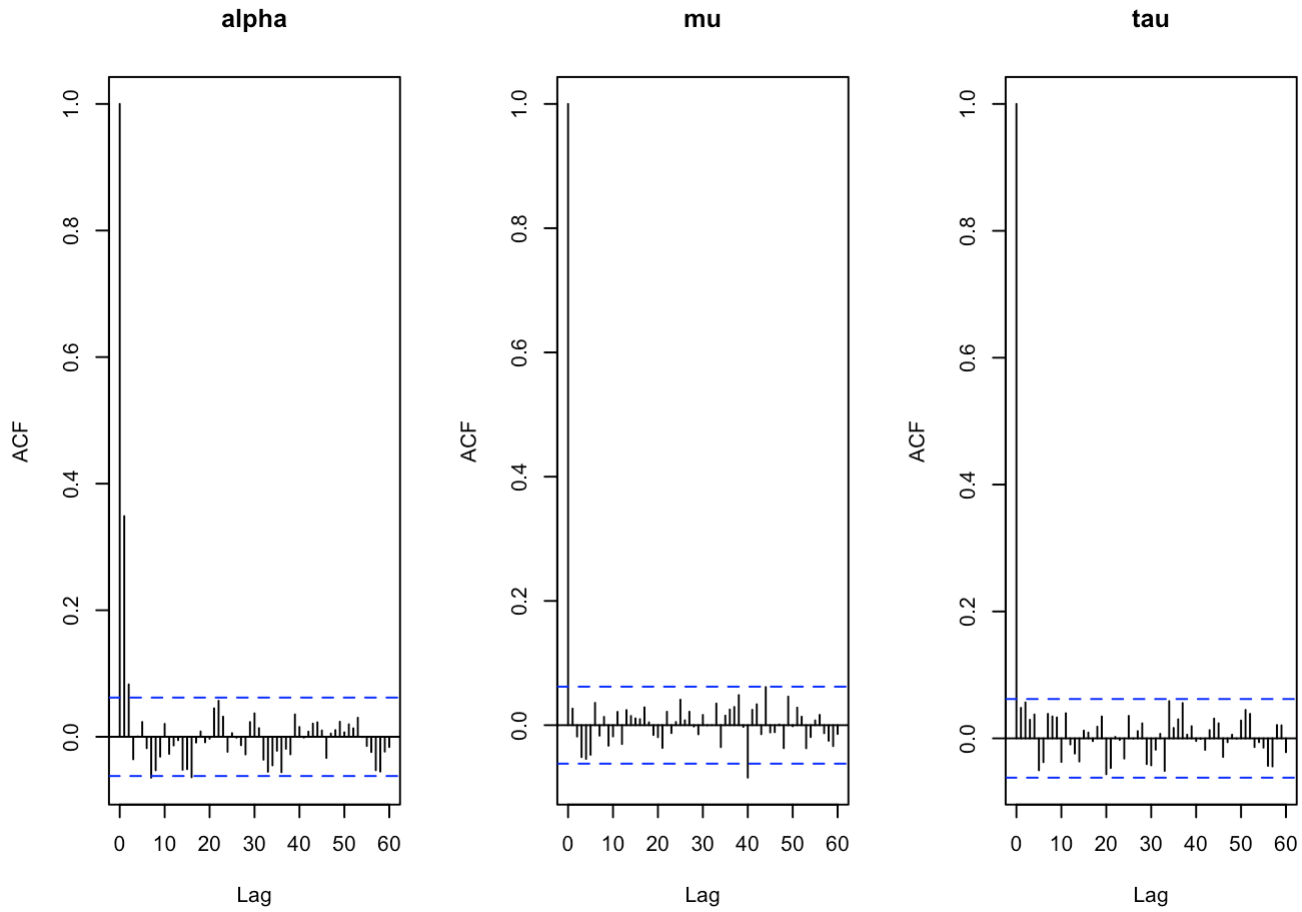
We now repeat this process for the other three chains.

```
mcmc_mat = as.matrix(samples[[1]])
par(mfrow=c(1,3))
for (j in 1:3) {
  acf(mcmc_mat[,j], lag.max=60, main=colnames(mcmc_mat)[j])
}
```



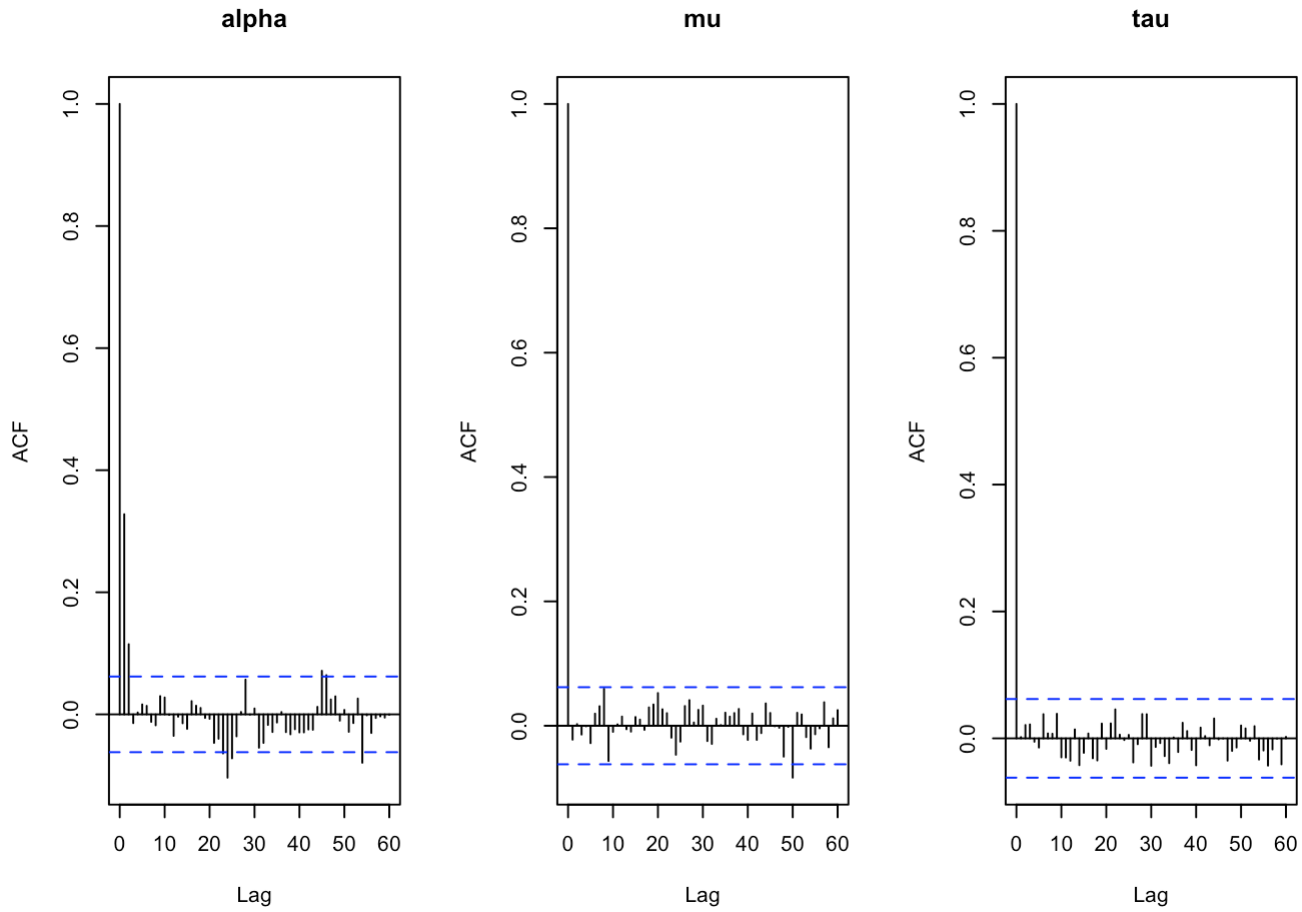
We now observe autocorrelation plots for chain 2. For α the ACF dips below the significance level at around time 3, for μ at around time 1, for τ at around time 2. Thus, as in chain 1, we thin to the largest of the three values adding a bit extra. Hence, chain 2 also suggests a thinning of 9.

```
mcmc_mat = as.matrix(samples[[2]])
par(mfrow=c(1,3))
for (j in 1:3) {
  acf(mcmc_mat[,j], lag.max=60, main=colnames(mcmc_mat)[j])
}
```



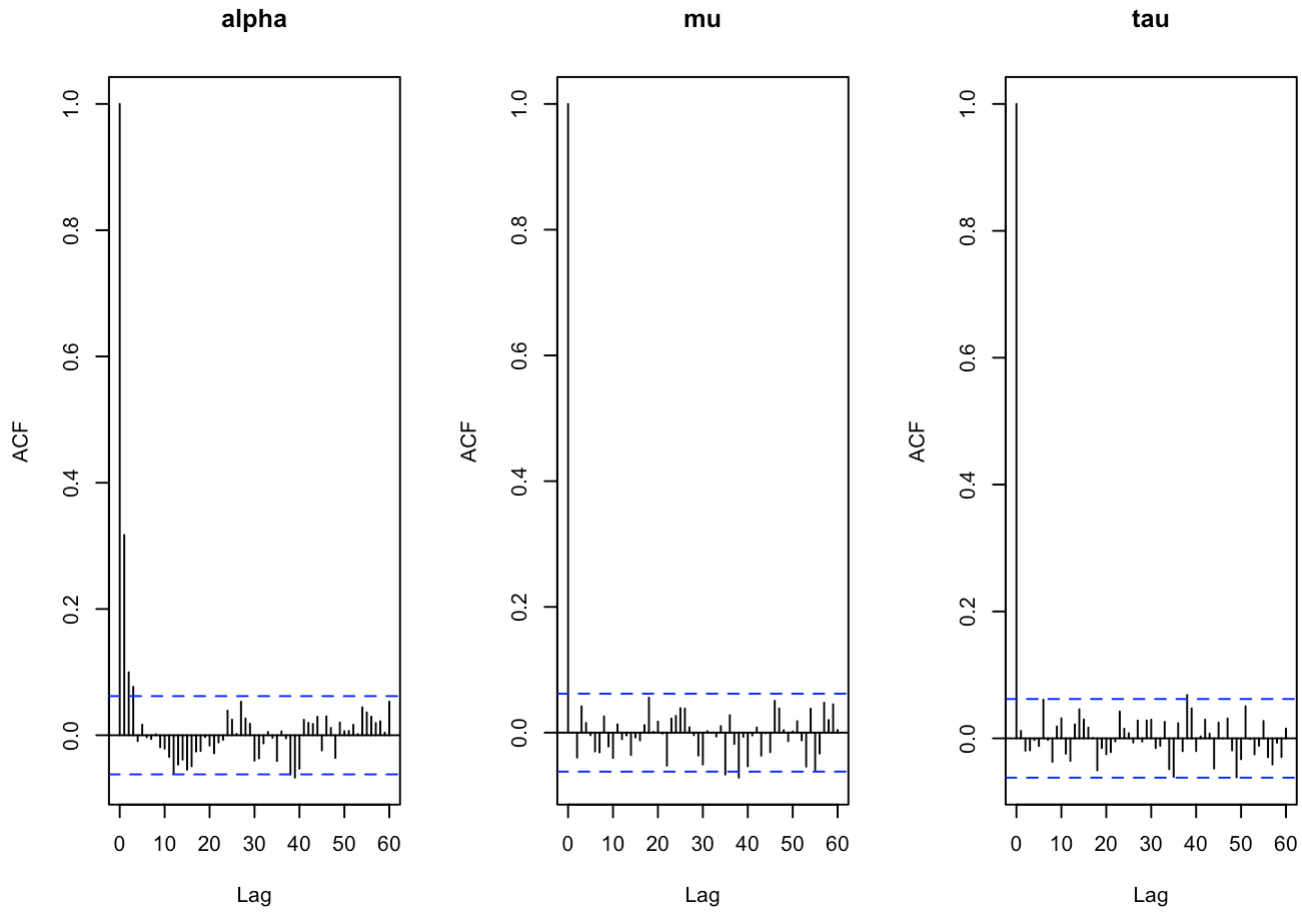
We now observe autocorrelation plots for chain 3. For α, the ACF dips below the significant level at around time 3, for μ at around time 1 and for τ at around time 2 - we thin to the largest of these three values adding a bit extra, hence chain 3 also suggests a thinning of 9.

```
mcmc_mat = as.matrix(samples[[3]])
par(mfrow=c(1,3))
for (j in 1:3) {
  acf(mcmc_mat[,j], lag.max=60, main=colnames(mcmc_mat)[j])
}
```



We now observe autocorrelation plots for chain 4. For α, the ACF dips below the significance level at around time 4, for μ at around time 1 and for τ at around time 1 - we thin to the largest of the three values and add a bit extra, thus chain 4 suggests a thinning of 10.

```
mcmc_mat = as.matrix(samples[[4]])
par(mfrow=c(1,3))
for (j in 1:3) {
  acf(mcmc_mat[,j], lag.max=60, main=colnames(mcmc_mat)[j])
}
```



Now chain 1, chain 2 and chain 3 all suggested a thinning of 9 whereas chain 4 suggested a thinning of 10 - to ensure we have no autocorrelation we thus decide to thin to every 10th iteration to give approximately uncorrelated samples.

We now re-initialise the model, however this time having 10*1000 = 10 000 iterations to ensure we still get 1000 samples (because of our thinning, we are only taking the sample of every 10 iterations).

```
# initialise the model with 4 MCMC chains (re-initialise so Markov Chain starts from
beginning - otherwise would take samples from where left off in old model)
model = jags.model(
    textConnection(modelstring),
    data=jags_data,
    n.chains=4
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 97
##    Unobserved stochastic nodes: 3
##    Total graph size: 367
##
## Initializing model
```

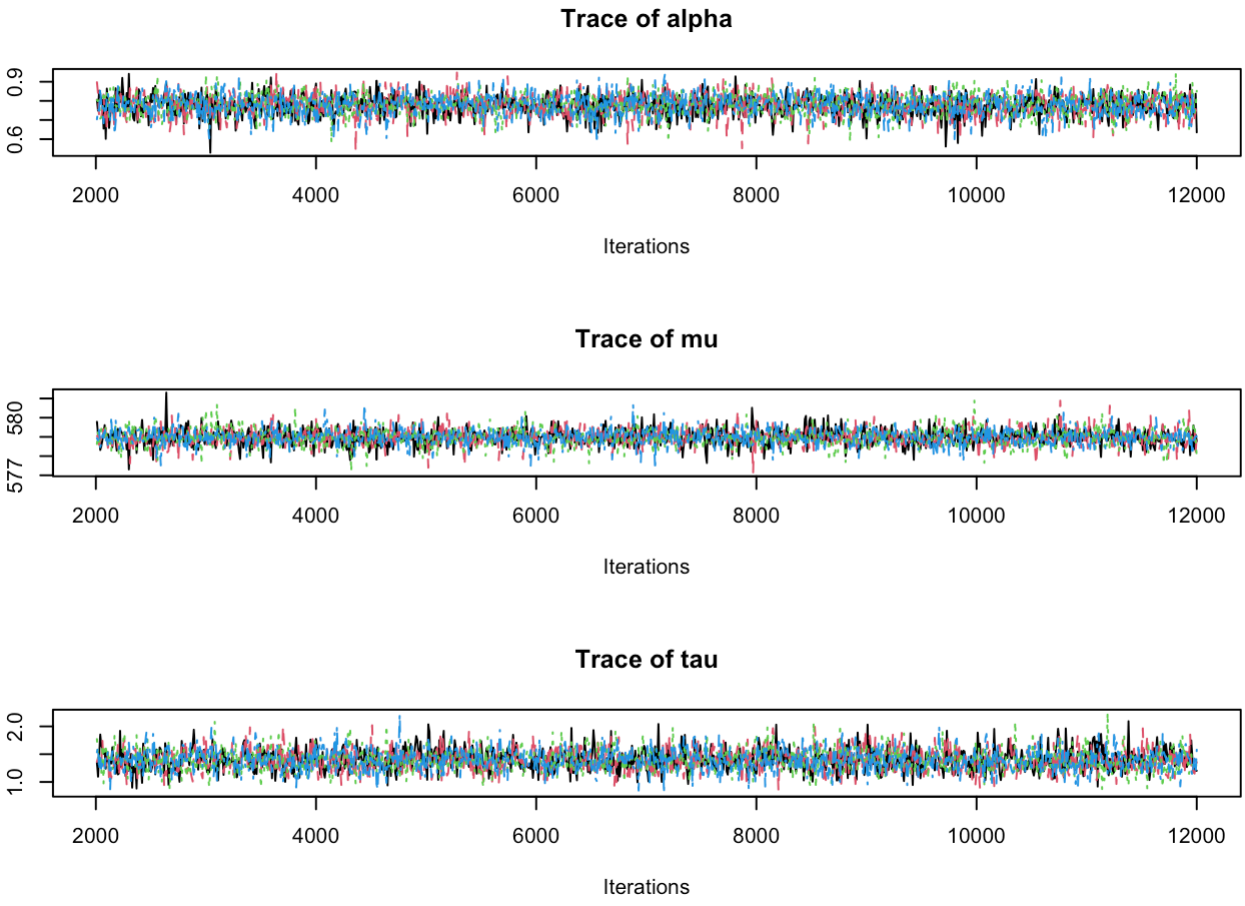
```
# perform burn-in and then MCMC with a thinning of 10 and 10 times as many iterations
to still ensure we have 1000 samples
update(model,n.iter=1000)
samples = coda.samples(model=model, variable.names=c("alpha", "mu", "tau"), n.iter=10
*1000, thin=10)
```

The trace plots are shown below.

The trace plot for alpha is now showing better mixing than previously.

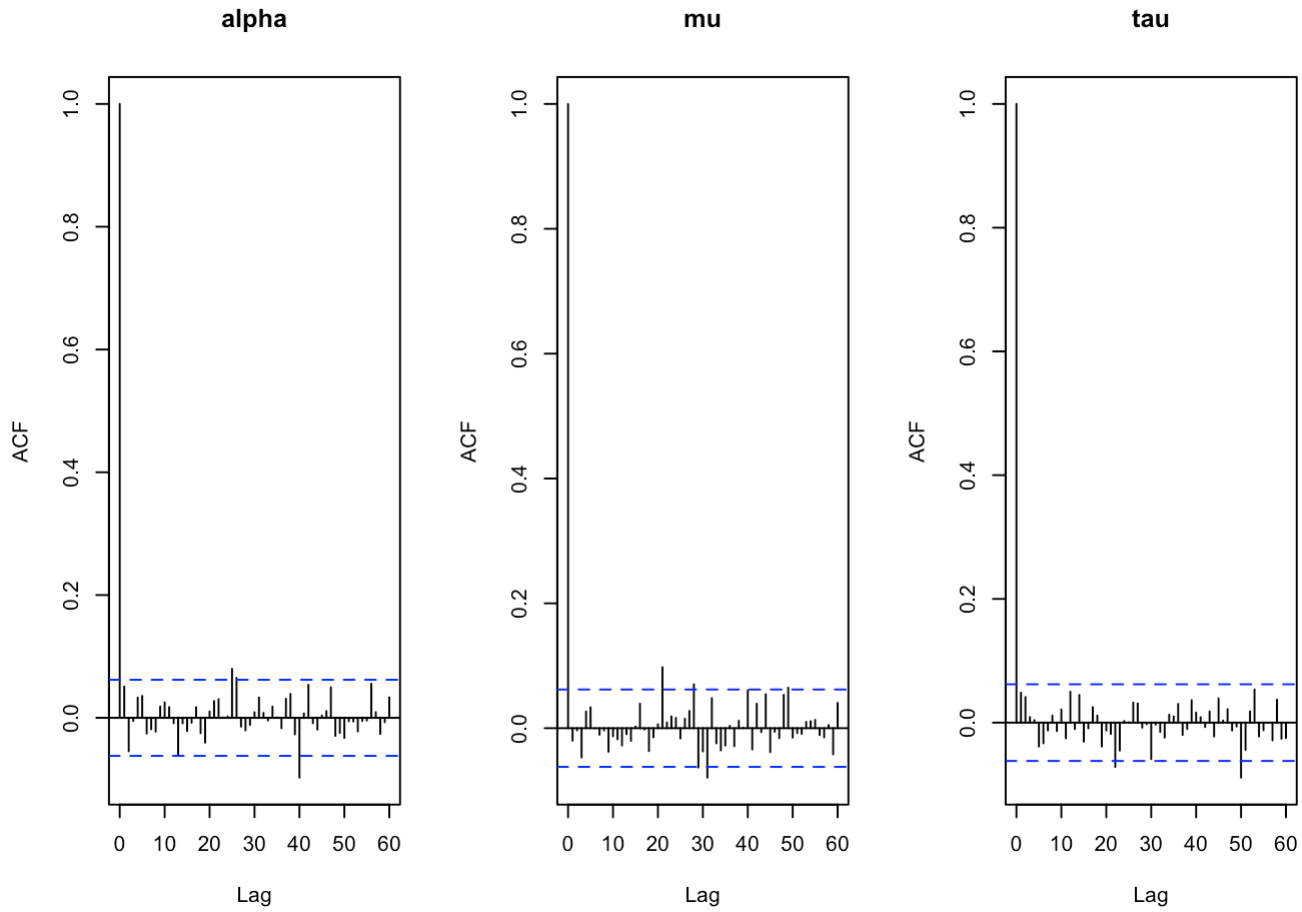
There has also been a significant improvement in the mixing of mu and the mixing of tau.

```
par(mfrow=c(3,1))
traceplot(samples)
```



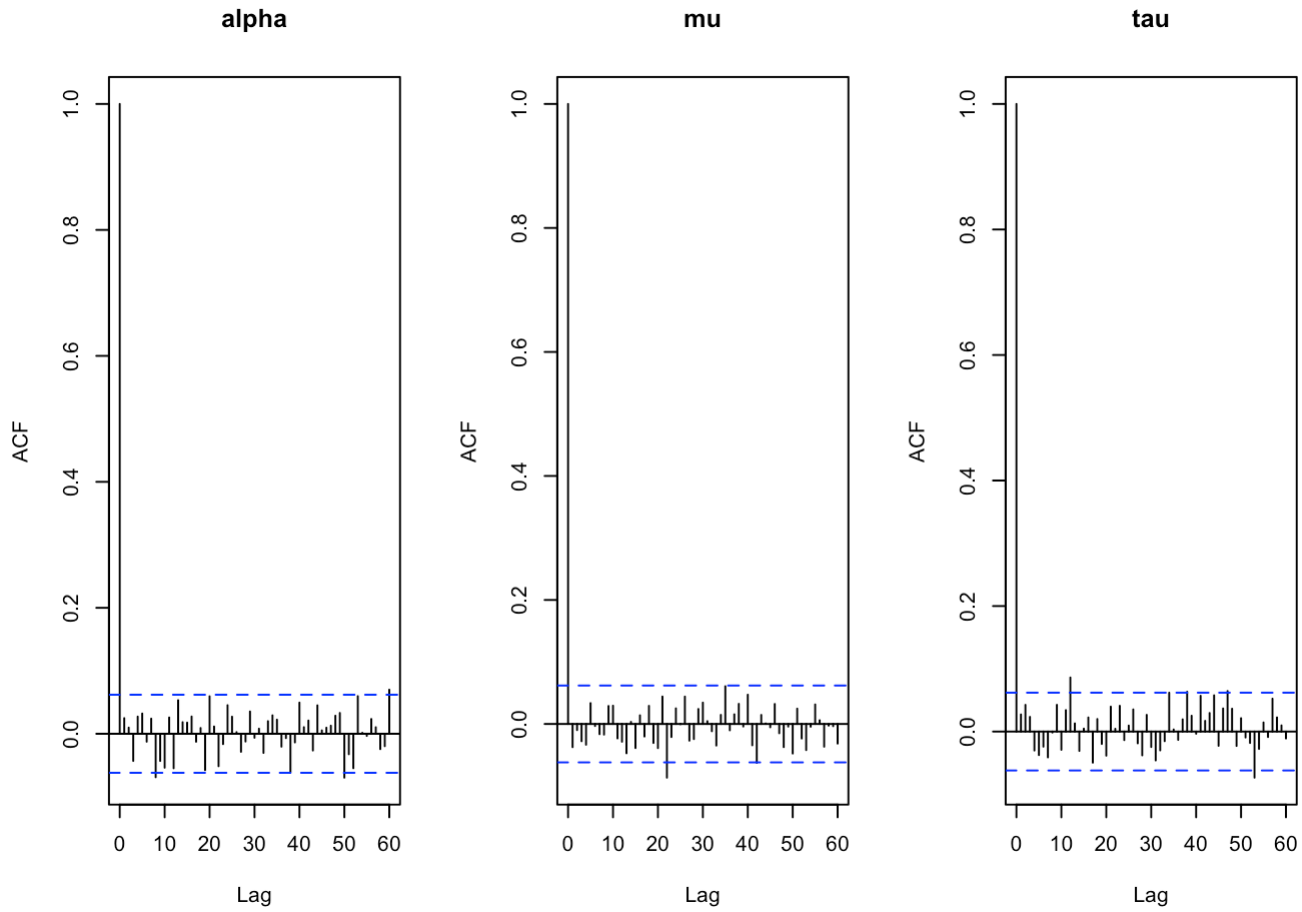
We now observe the autocorrleation plots for all 4 chains, beginning with chain 1. We see there is no evidence of autocorrleation in this thinned chain.

```
mcmc_mat = as.matrix(samples[[1]])
par(mfrow=c(1,3))
for (j in 1:3) {
  acf(mcmc_mat[,j], lag.max=60, main=colnames(mcmc_mat)[j])
}
```



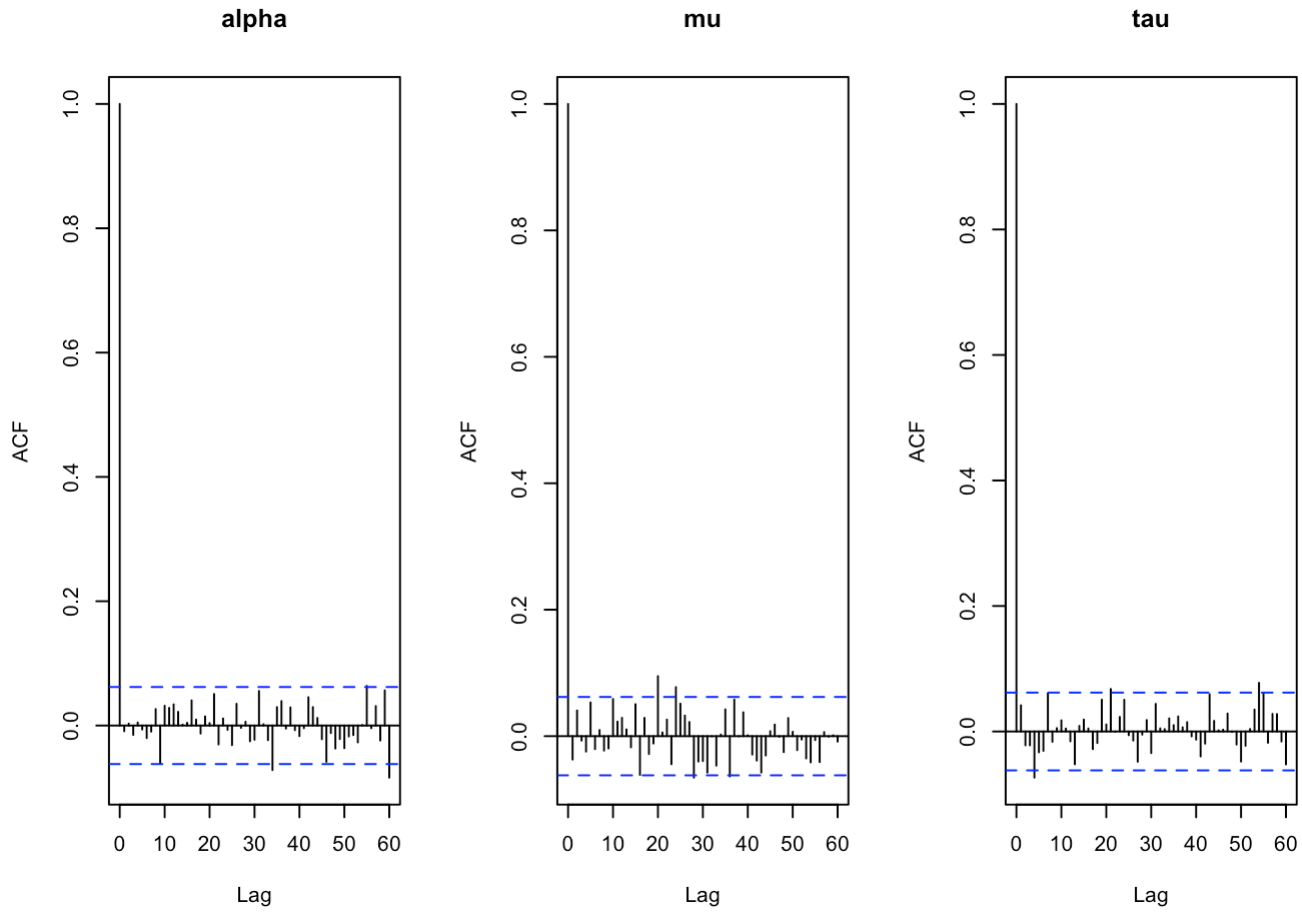
For chain 2, again there is no evidence of autocorrelation in the thinned chain.

```
mcmc_mat = as.matrix(samples[[2]])
par(mfrow=c(1,3))
for (j in 1:3) {
  acf(mcmc_mat[,j], lag.max=60, main=colnames(mcmc_mat)[j])
}
```



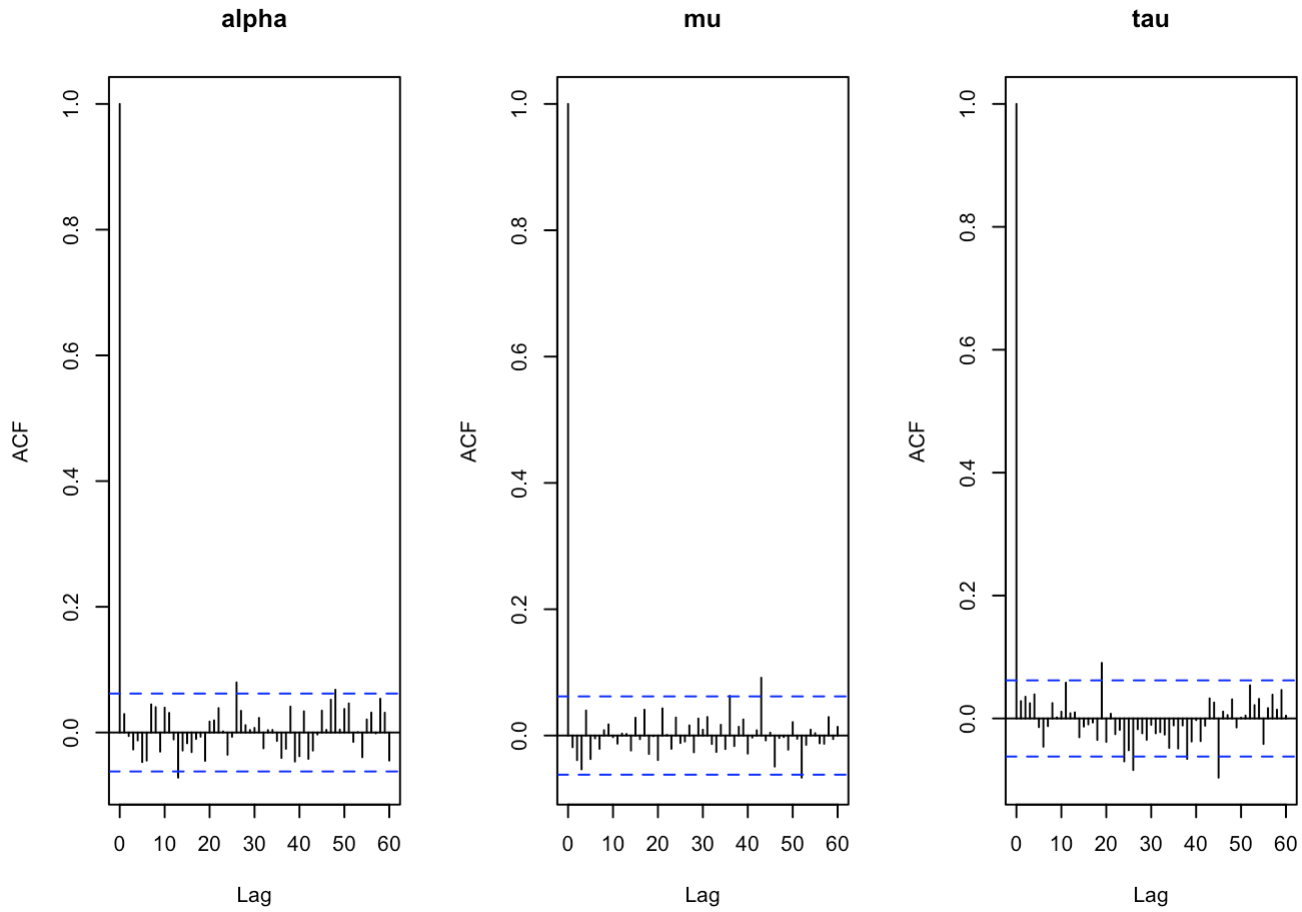
For chain 3, again there is no evidence of autocorrelation in the thinned chain.

```
mcmc_mat = as.matrix(samples[[3]])
par(mfrow=c(1,3))
for (j in 1:3) {
  acf(mcmc_mat[,j], lag.max=60, main=colnames(mcmc_mat)[j])
}
```



For chain 4, again there is no evidence of autocorrelation in the thinned chain.

```
mcmc_mat = as.matrix(samples[[4]])
par(mfrow=c(1,3))
for (j in 1:3) {
  acf(mcmc_mat[,j], lag.max=60, main=colnames(mcmc_mat)[j])
}
```



Thus, the diagnostic plots of the MCMC output with a thinning of 10 demonstrate mixing to perform well and no autocorrelation thus our tuning choices lead to our MCMC output behaving well.

Question 3

We calculate posterior expectations and 95% HPD intervals for each parameter using the MCMCvis library

The code below then gives us the following:

The posterior expectation for α is 0.7776212 The 95% HPD interval for α is [0.6694061, 0.8933991]

The posterior expectation for μ is 578.9732054 The 95% HPD interval for μ is [578.1396635, 579.9113071]

The posterior expectation for τ is 1.3905879 The 95% HPD interval for τ is [1.0103376, 1.7684983]

```
library(MCMCvis)
MCMCsummary(samples, HPD=TRUE, hpd_prob =0.95)
```

```
##              mean         sd    95%_HPDL    95%_HPDU Rhat n.eff
## alpha   0.7780546 0.05868979   0.6644593   0.8916884    1  4011
## mu    578.9802536 0.43679989 578.1539897 579.9057623    1  4000
## tau     1.3943551 0.20163189   0.9945270   1.7867874    1  3907
```

Question 4

To get the posterior probabillity of $\alpha > 0.9$, we find how many of the samples in the MCMC output for α are greater than 0.9. We then divide this by the total number of samples for α.

The code below does the following.

```
alpha_samples = as.matrix(samples[, 'alpha']) # get all samples of alpha from MCMC an
alysis
N = 4000 # the total number of samples (each chain had 1000 samples, we have 4 chains
thus 4000 samples)
val_interest = 0.9 # value of interest
num_greater = 0 # stores the number of samples greater than 0.9

for (i in 1:N) { # for each sample, increments num_greater if the sample > 0.9
  if (alpha_samples[i] > val_interest) {
    num_greater = num_greater + 1
  }
}

posterior_prob = num_greater / N # posterior probabillity

prior_prob = 1 - pbeta(val_interest, 2, 5) # prior density of alpha > 0.9

paste("Posterior probabillity for alpha > 0.9 is ", posterior_prob)
```

```
## [1] "Posterior probabillity for alpha > 0.9 is  0.01275"
```

```
paste("Prior probabillity for alpha > 0.9 is ", prior_prob)
```

```
## [1] "Prior probabillity for alpha > 0.9 is  5.50000000000272e-05"
```

Thus, we observe the prior probability of $\alpha > 0.9$ is very small. After observing the data, the posterior probability of $\alpha > 0.9$ increases, suggesting the data provides little but some evidence for $\alpha > 0.9$.

Question 5

The code below shows us the correlation matrix and the scatter plots for our parameters.

The correlation matrix suggests α and μ to have a very small but negative correlation, α and τ to have a small but positive correlation and μ and τ to have a small but positive correlation.

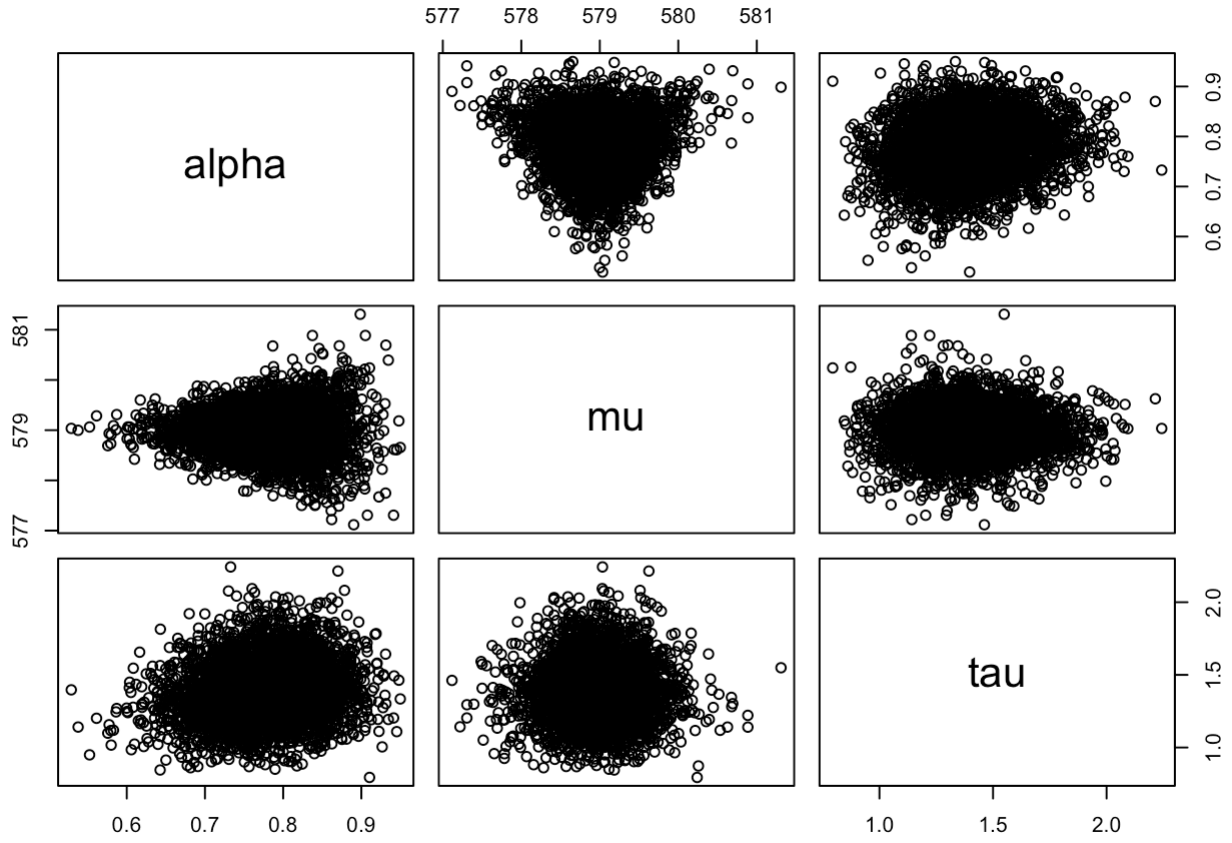
This observation can also be seen in the scatter plots: for example, as μ increases most values of α very slowly drifts downwards. Similarly, as τ increases, most values of α slowly drifts upwards and as τ increases, most values of μ slowly drifts upwards.

This suggests that, after observing the data, we find the parameters to have some dependency between them and not be truly independent. This dependency is very small (close to 0) for α and μ however slightly more prevalent for μ and τ and most prevalent for α and τ.

```
samples_matrix <- as.matrix(samples)
cor(samples_matrix)
```

```
##             alpha           mu          tau
## alpha  1.00000000 -0.020263644  0.140548084
## mu    -0.02026364  1.000000000 -0.001280728
## tau    0.14054808 -0.001280728  1.000000000
```

```
pairs(samples_matrix)
```



Question 6.

Begin by defining the model. We add a new parameter X_{99} to the model.

```
modelstring = "model {
  alpha ~ dbeta(2, 5)
  mu ~ dexp(0.01)
  tau ~ dexp(10)
  sd = pow(tau, -0.5)
  for (i in 2:N) {
    x[i] ~ dnorm(mu + alpha*(x[i-1]-mu) , tau)
  }
  x_99 ~ dnorm(mu + alpha*(x[98]-mu), tau)
}"
```

Get the data.

```
lake_huron = LakeHuron # gets the data
jags_data = list(x=lake_huron, N=98)
```

Run burn-in for 1000 observations and then perform MCMC for 1000 time steps. We perform MCMC with a thinning of 10 as done previously, because Q2 showed us without this thinning there is some dependency between each time step for each parameter (α, μ and τ) and without this thinning mixing could be improved (as shown in the trace plots and autocorrelation plots in Q2). We do 10*1000 iterations to ensure we get 1000 samples (because we only use every 10th sample).

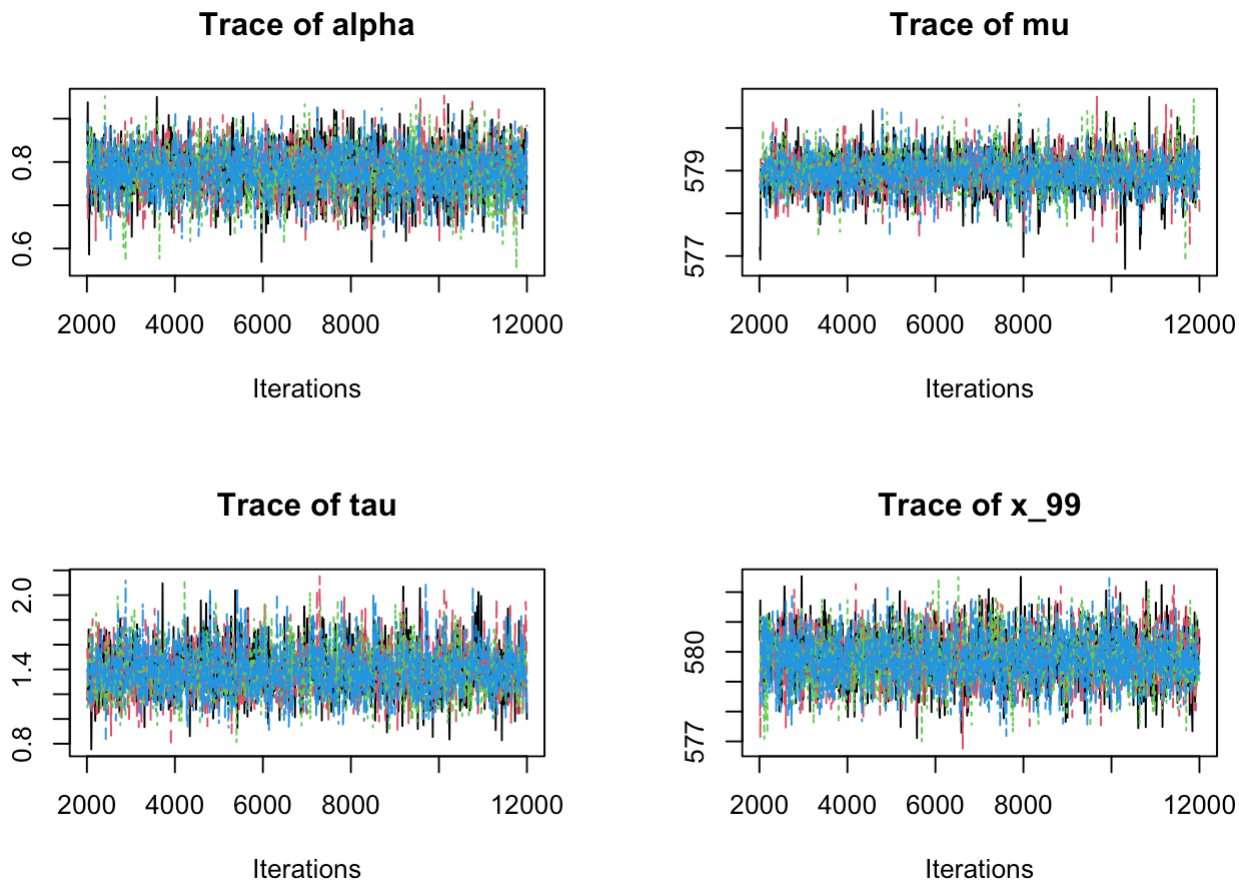
```
# initialize the model with 4 MCMC chains
model = jags.model(
    textConnection(modelstring),
    data=jags_data,
    n.chains=4
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 97
##    Unobserved stochastic nodes: 4
##    Total graph size: 368
##
## Initializing model
```

```
# perform burn-in and then MCMC with a thinning of 10 and 10 times as many iterations
to still ensure we have 1000 samples
update(model,n.iter=1000)
samples = coda.samples(model=model, variable.names=c("alpha", "mu", "tau", "x_99"),
n.iter=10*1000, thin=10)
```

We observe the trace plots of each parameter to ensure we converge to the posterior distribution - the trace plots all show a ‘hairy caterpillar’ pattern and thus we can be fairly confident in our convergence to the posterior distribution.

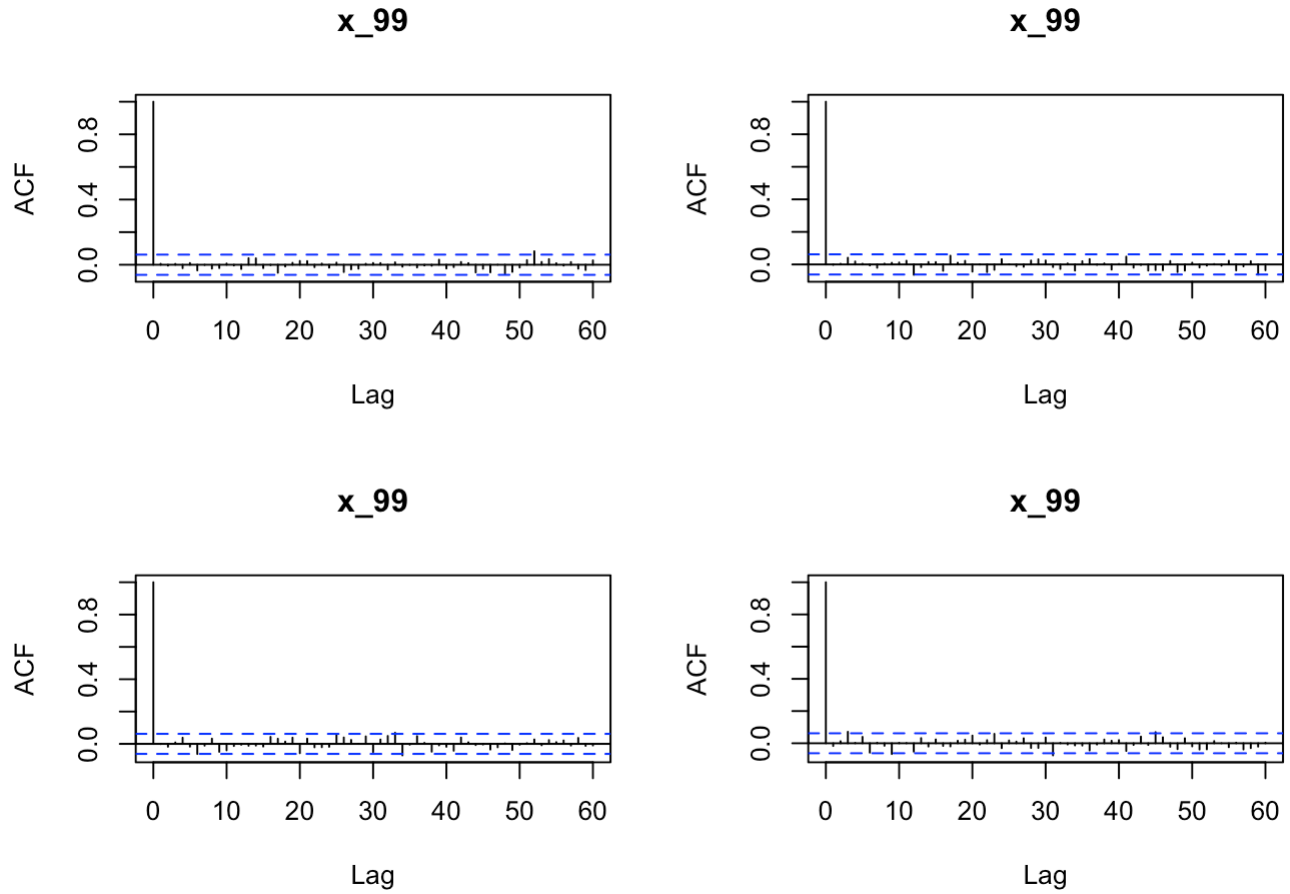
```
par(mfrow=c(2,2))
x_99_samples <- as.matrix(samples[, "x_99"])
traceplot(samples)
```



We also double check if there is any autocorrelation for X_{99} - the autocorrelation plots indeed show no dependency and no evidence of autocorrelation in each chain.

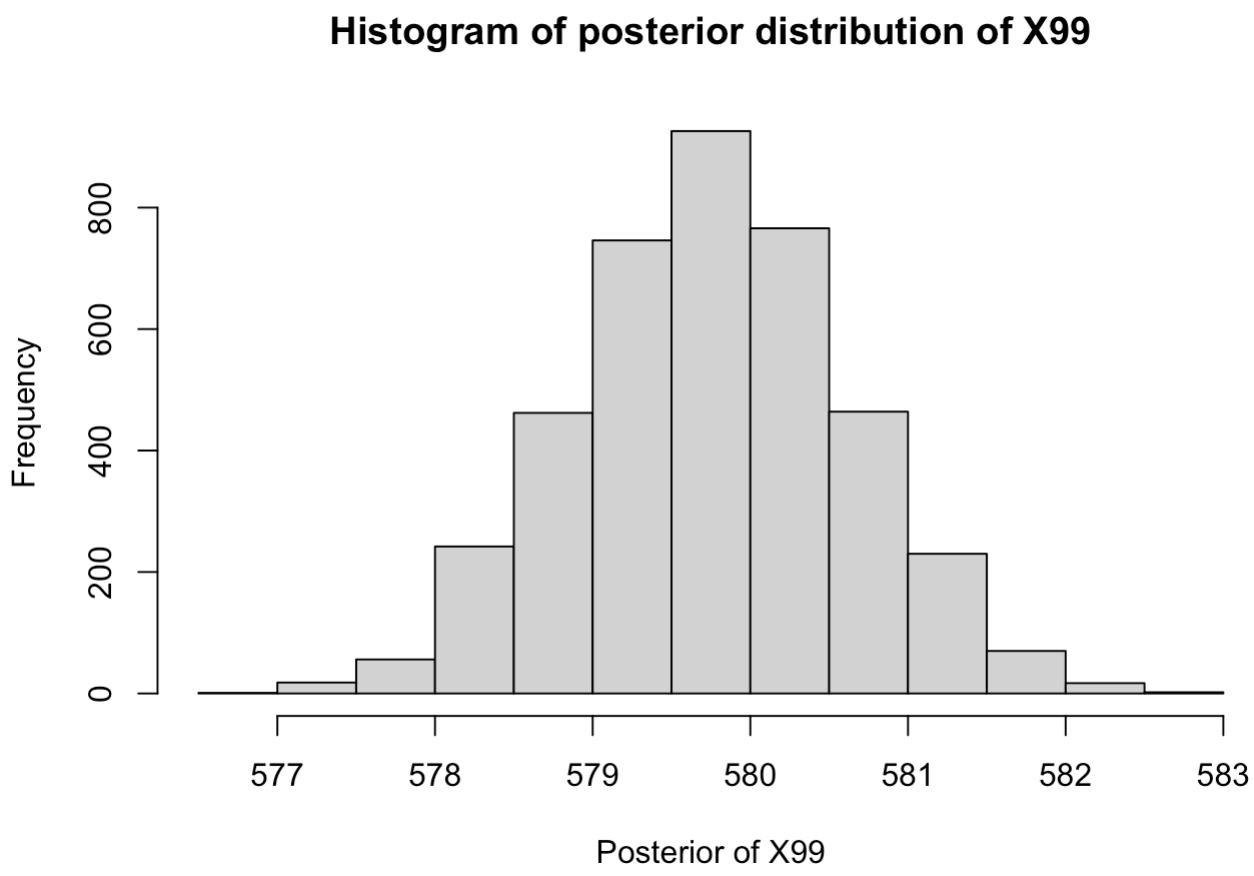
```
mcmc_mat1 = as.matrix(samples[[1]])
mcmc_mat2 =
par(mfrow=c(2,2))
for (i in 1:4) {
  mcmc_mat = as.matrix(samples[[i]])
  acf(mcmc_mat[,4], lag.max=60, main=colnames(mcmc_mat)[4])

}
```



We now plot the histogram of X_{99} under the posterior. We observe a larger frequency of values between 579.5 and 580 with most values between 579 and 580.5.

```
x_99_samples <- as.matrix(samples[, "x_99"])
hist(x_99_samples, xlab="Posterior of X99", main="Histogram of posterior distribution
of X99")
```


Histogram of posterior distribution of X99
Frequency
0
200
400
600
800
577
578
579
580
581
582
583
Posterior of X99