# Bayesian HW5

*Kerui Cao*

*11/10/2019*

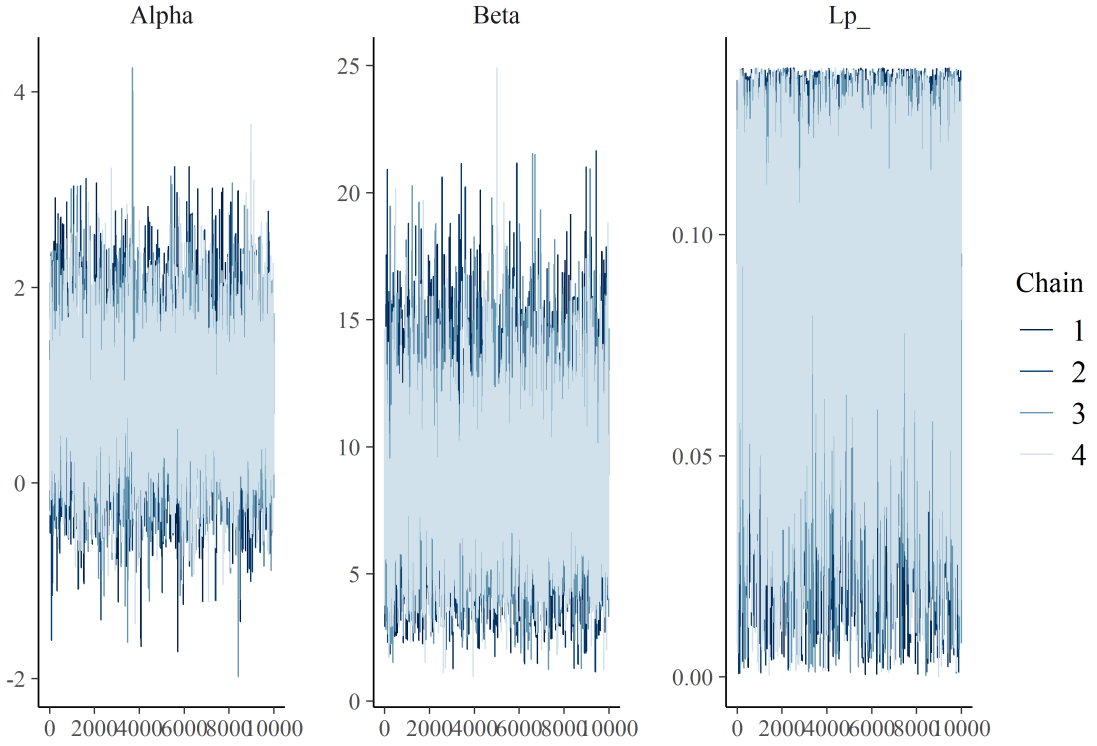
## BDA Problem 11.2

Metropolis algorithm: Replicate the computations for the bioassay example of Section 3.7 using the Metropolis algorithm. Be sure to define your starting points and your jumping rule. Compute with log-densities (see page 261). Run the simulations long enough for approximate convergence.

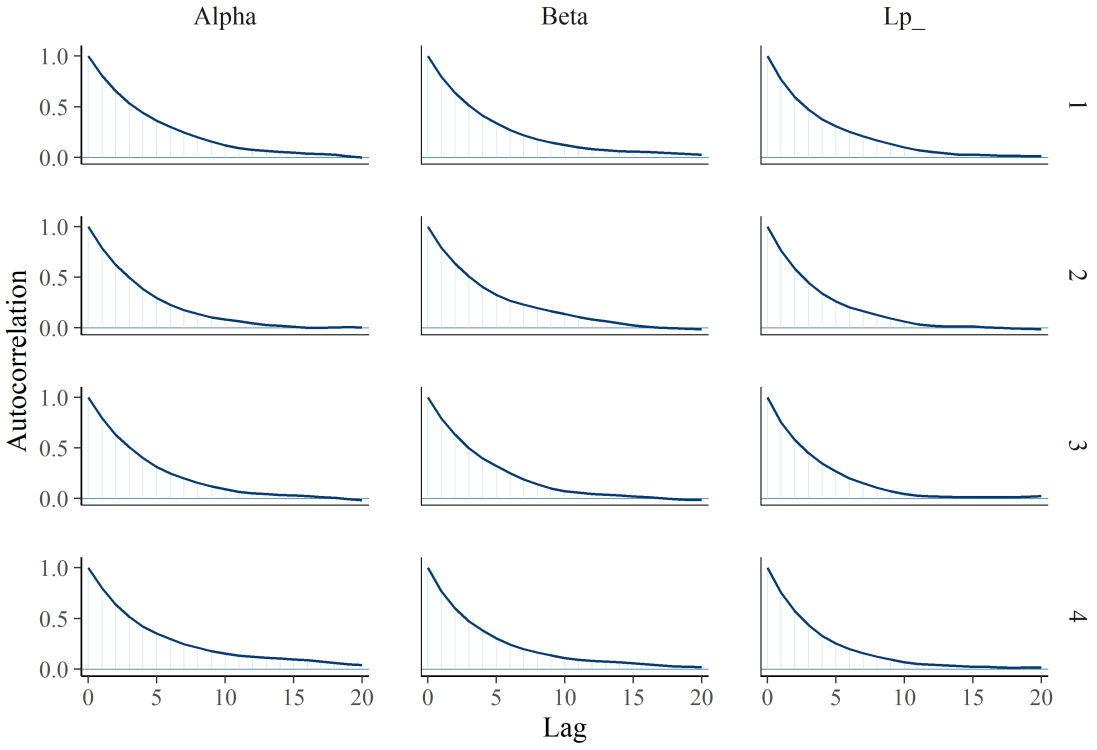
```
a.c = 0.8
b.c = 7.7
a.sd = 1
b.sd = 4.9

mcmc_array <- function (ns, nchains = 1, params) {
  nparams <- length(params)
  array(dim = c(ns, nchains, nparams),
        dimnames = list(iterations = NULL,
                        chains = paste0("chain", 1:nchains),
                        parameters = params))
}

data = data.frame("y" = c(0,1,3,5),
                  "n" = c(5,5,5,5),
                  "x" = c(-0.86,-0.3,-0.05,0.73))

in.lo = function(al,be){
  the = al + be * data$x
  the = invlogit(the)
  return(the)
}

po = function(al,be){
  the = in.lo(al=al,be=be)
  sum.cho = sum(lchoose(n = data$n,k = data$y))
  p2 = sum(data$y*log(the))
  p3 = sum(log(1-the)*(data$n-data$y))
  lopo = exp(sum.cho + p2 + p3)
  return(lopo)
}

nc = 4
ns = 10000

cs = c("Alpha","Beta","Lp_")
sims = mcmc_array(ns,nchains = nc, params = cs)
for(j in 1:nc){
  for(i in 1:ns){
  a.s = rnorm(n = 1,mean = a.c,sd = a.sd)
  b.s = rnorm(n = 1,mean = b.c,sd = b.sd)
```

```
  up = po(al = a.s,be = b.s)*dnorm(a.s,mean = 0.8,sd = 1)*dnorm(b.s,mean = 7.7,sd = 4.9)
  do = po(al = a.c,be = b.c)*dnorm(a.c,mean = 0.8,sd = 1)*dnorm(b.c,mean = 7.7,sd = 4.9)
  r = up/do
  a = runif(1)
  if(r >a){
    a.c = a.s
    b.c = b.s
  }
  post = po(al = a.c,be = b.c)
  sims[i,j,] = c(a.c,b.c,post)
  }
}
```

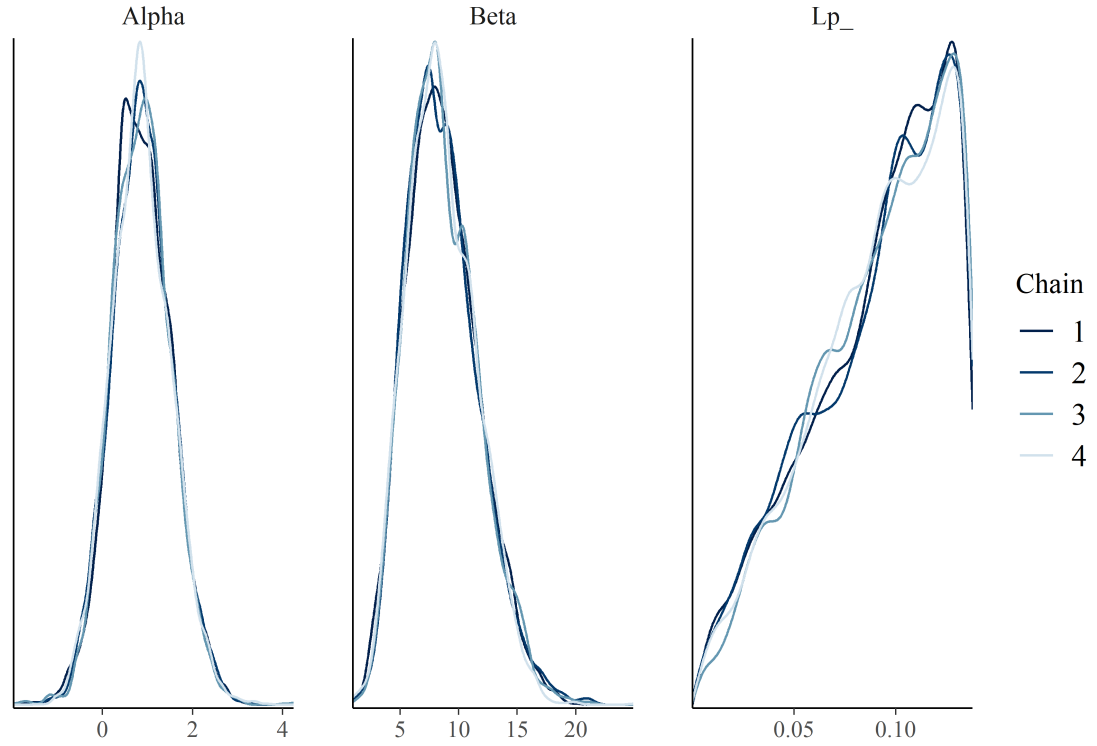
```
mcmc_trace(sims, cs)
```



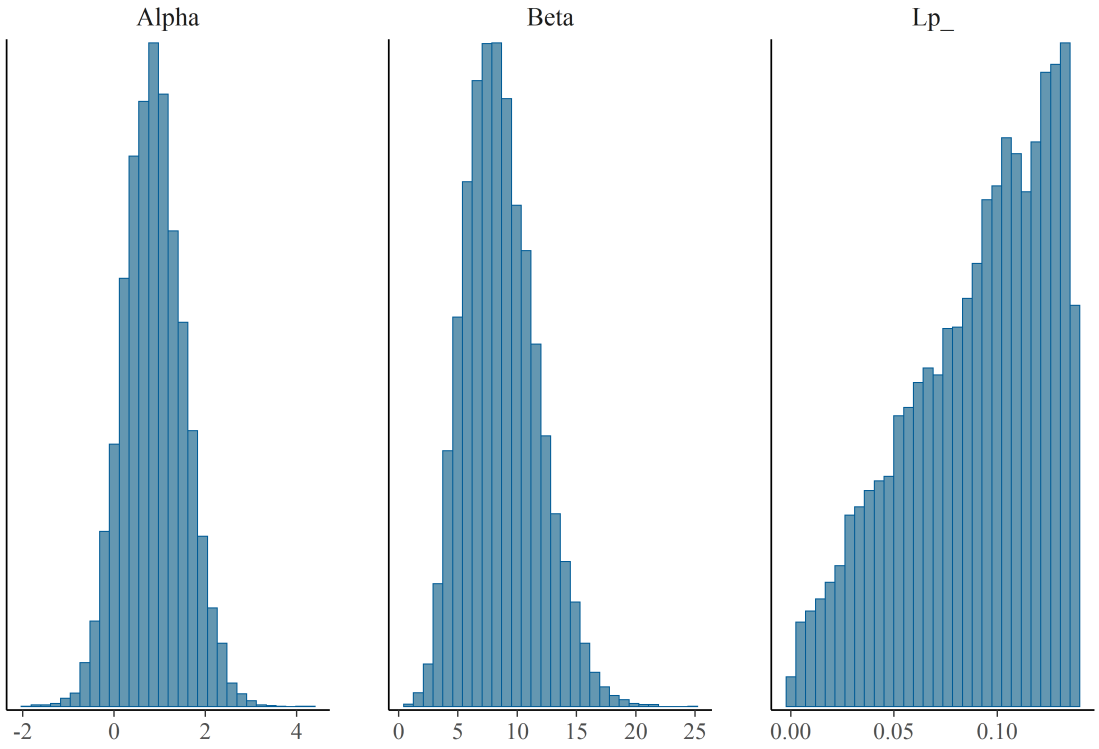

```
mcmc_acf(sims, cs)
```




```
mcmc_dens_overlay(sims, cs)
```



```
mcmc_hist(sims, cs)
```



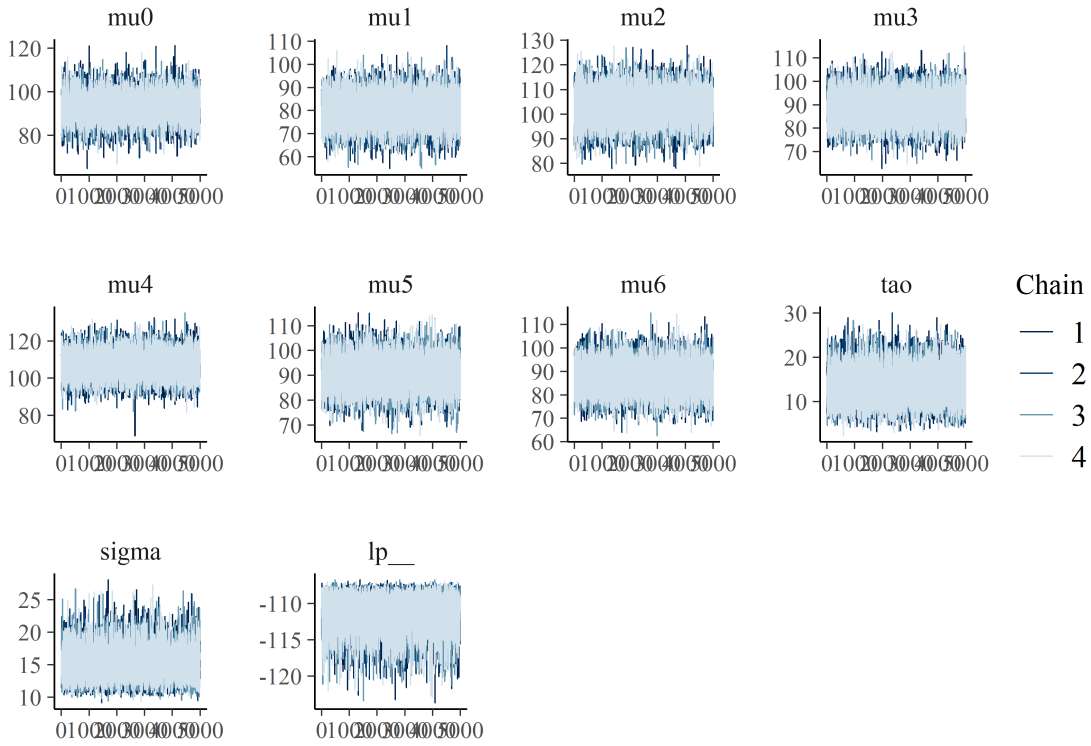

# BDA Problem 11.3

```
mo1 = stan_model("machine.stan")
```

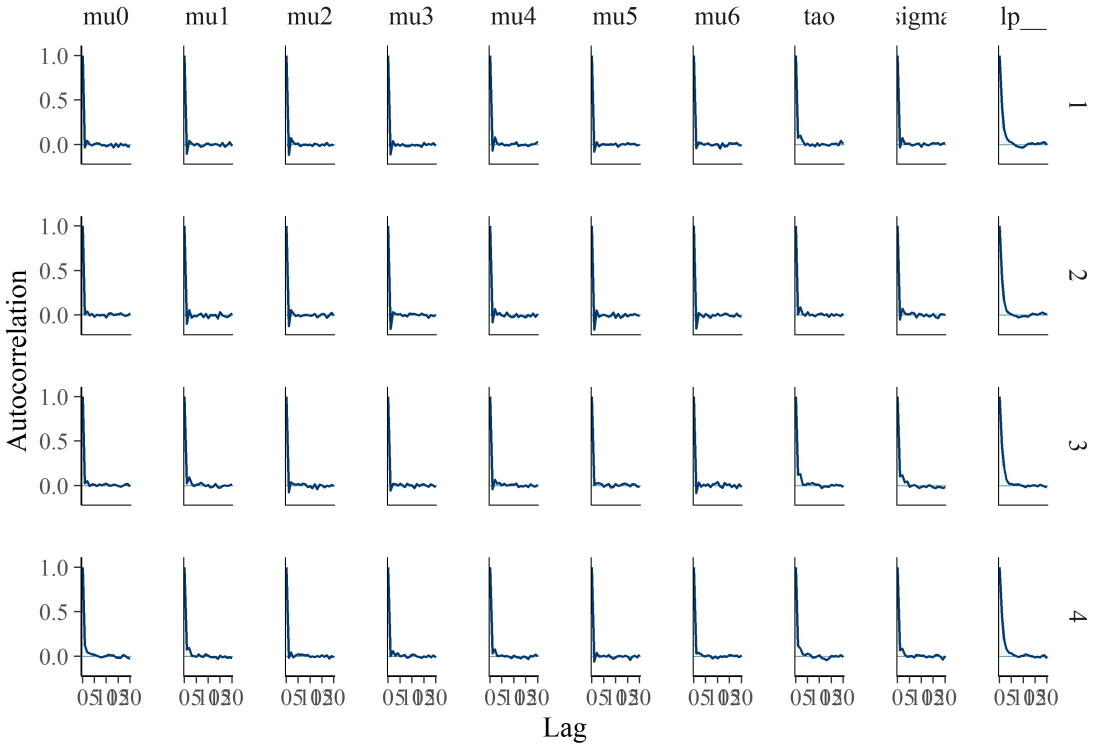
```
monitor(sf)
```

```
## Inference for the input samples (4 chains: each with iter = 10000; warmup = 0):
##
##           Q5    Q50    Q95   Mean  SD  Rhat Bulk_ESS Tail_ESS
## mu0     83.7   93.1  102.6   93.1 5.8     1    16634    14411
## mu1     69.7   79.9   90.6   80.0 6.4     1    18227    12791
## mu2     92.9  103.2  113.3  103.2 6.2     1    19894    13609
## mu3     78.9   89.0   99.0   89.0 6.1     1    20956    14693
## mu4     96.7  107.5  118.0  107.4 6.5     1    17763    12907
## mu5     81.0   90.7  100.7   90.7 6.1     1    21780    13557
## mu6     77.6   87.6   97.8   87.6 6.2     1    21021    14332
## tao      7.7   13.3   20.1   13.5 3.8     1    13141    11789
## sigma   11.8   14.9   19.3   15.1 2.3     1    16566    13162
## lp__  -115.6 -111.0 -108.2 -111.3 2.3     1     7979    12525
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

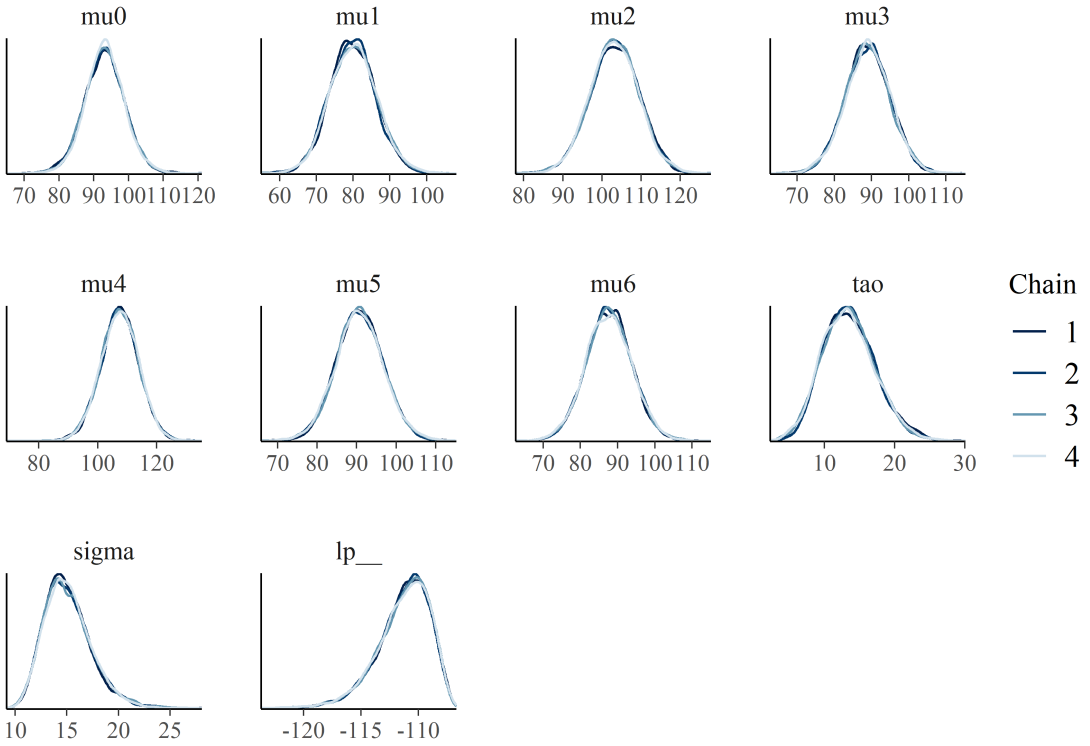
```
sims <- as.array(sf)
params <- c(paste0("mu", 0:6), "tao", "sigma", "lp__")
mcmc_trace(sims, params)
```



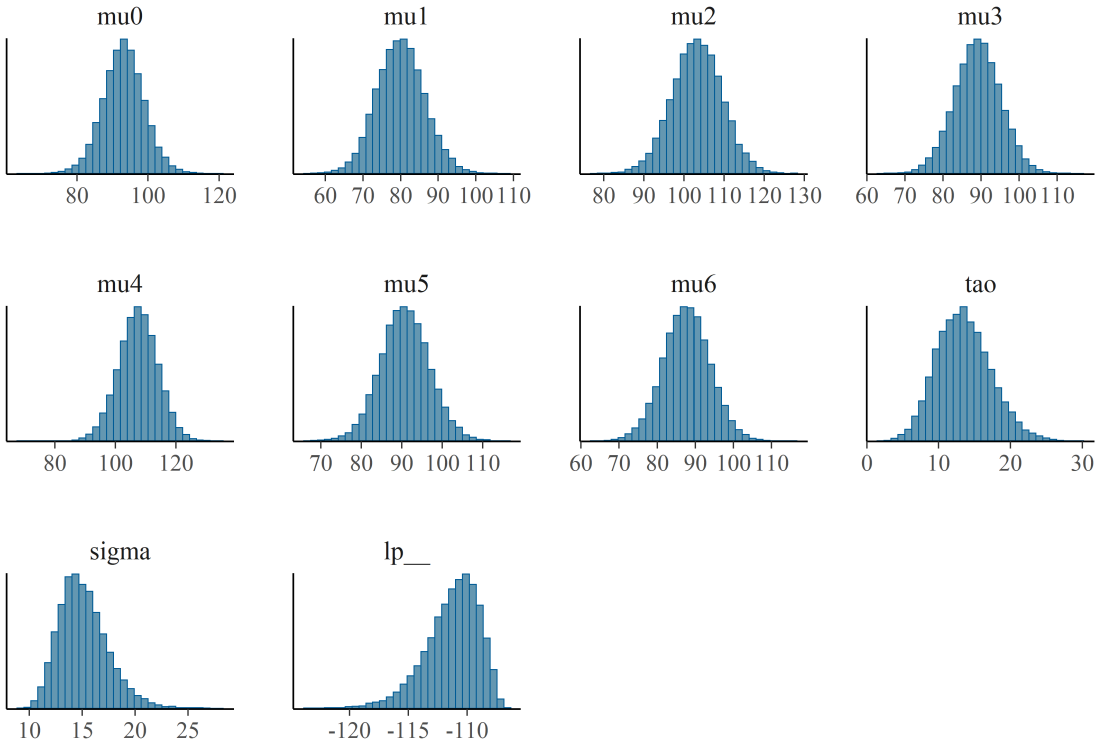

```
mcmc_acf(sims, params)
```




```
mcmc_dens_overlay(sims, params)
```



```
mcmc_hist(sims, params)
```




## Question 3

```
data = data.frame("y" = c(13,52,6, 40, 10, 7, 66, 10, 10, 14, 16, 4,
                          65, 5, 11, 10, 15, 5, 76, 56, 88, 24, 51, 4,
                          40, 8, 18, 5, 16, 50, 40, 1, 36, 5, 10, 91,
                          18, 1, 18, 6, 1, 23, 15, 18, 12, 12, 17, 3))
```

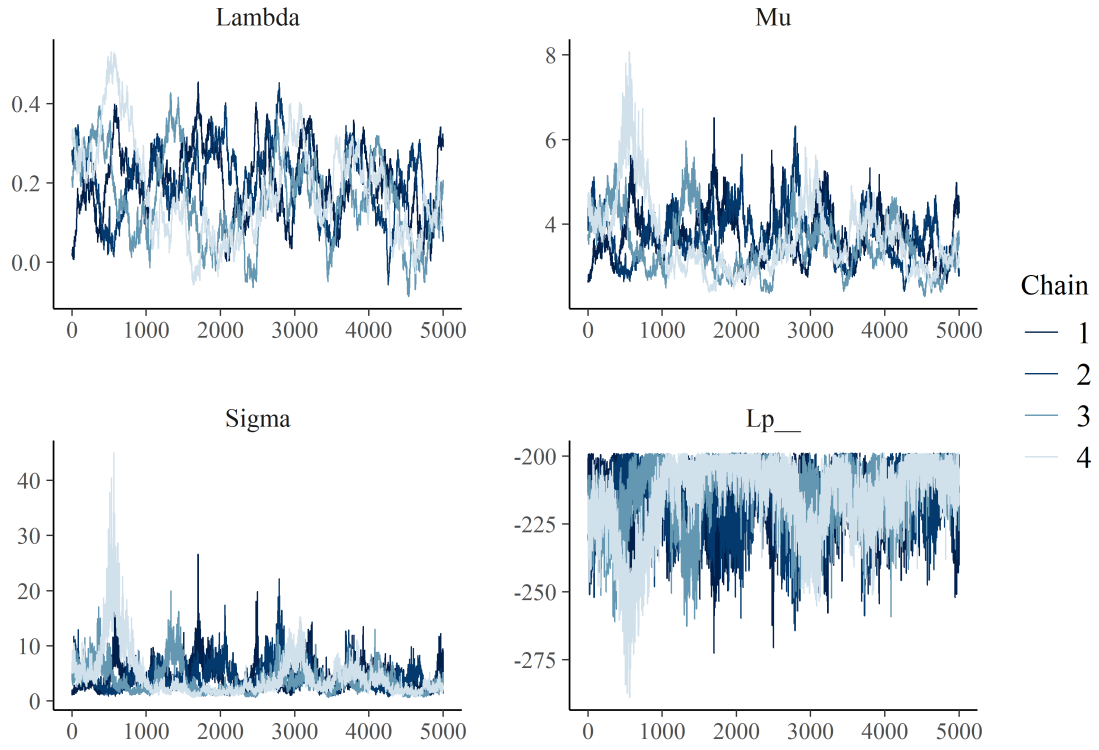
```
y = data$y
mcmc_array <- function (ns, nchains = 1, params) {
  nparams <- length(params)
  array(dim = c(ns, nchains, nparams),
        dimnames = list(iterations = NULL,
                        chains = paste0("chain", 1:nchains),
                        parameters = params))
}

nc = 4
ns = 10000
cs = c("Lambda","Mu","Sigma","Lp__")

la.c = 0.5
mu.c = 3.8
si.c = 5.1

la.sd = 0.01

n = length(y)

sims = mcmc_array(ns,nchains = nc, params = cs)

wi = function(la){
  y = data$y
  if(abs(la)<0.005){
    w = log(y)
  } else{
    w = (y^la-1)/la
  }
  return(w)
}

w.mu = function(la,mu){
  w = wi(la = la)
  re = 0-sum((w-mu)^2)
  return(re)
}

mean.mu = function(la){
  w = wi(la = la)
  return(mean(w))
}

si.n = function(si){
  return(si^(-(n+1)/2))
}
for(j in 1:nc){
  for(i in 1:ns){
```

```
    mu.c = rnorm(n = 1,mean = mean.mu(la = la.c),sd = si.c/n)
    si.c = 1/rgamma(n = 1,shape = (n-1)/2,rate = -w.mu(la = la.c,mu = mu.c)/2)
    la.s = rnorm(n = 1,mean = la.c,sd = la.sd)
    up = sum((la.s-1)*log(y))+w.mu(la = la.s,mu = mu.c)/(2*si.c)
    down = sum((la.c-1)*log(y))+w.mu(la = la.c,mu = mu.c)/(2*si.c)
    r1 = exp(up-down)
    if(r1 >= runif(1)){la.c = la.s}else{la.c = la.c}
    lp = -0.5*log(si.c) + sum(log(dnorm(wi(la = la.c),mean = mu.c,sd = si.c))) + sum((la.c-1)*log(y))
    sims[i,j,] = c(la.c,mu.c,si.c,lp)
  }
}
```

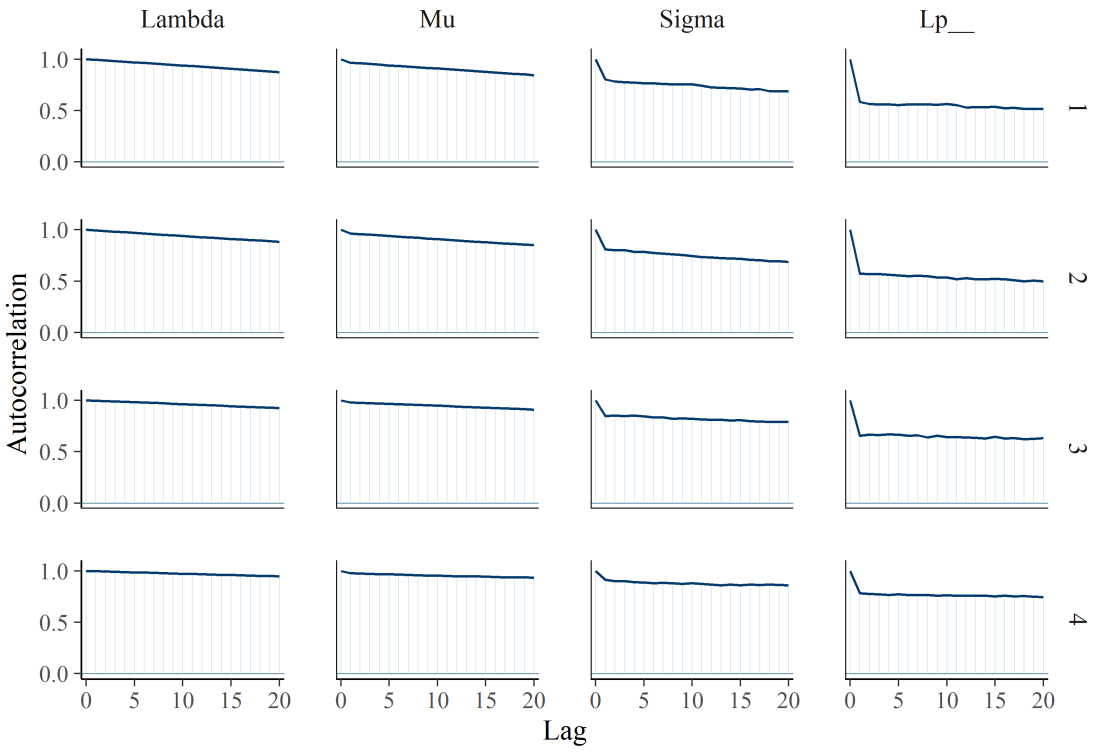
```
monitor(sims[,c(1,3),])
```

```
## Inference for the input samples (2 chains: each with iter = 10000; warmup = 5000):
##
##             Q5    Q50    Q95   Mean   SD  Rhat Bulk_ESS Tail_ESS
## Lambda     0.0    0.2    0.3    0.2  0.1  1.05       38       70
## Mu         2.7    3.5    4.8    3.6  0.6  1.04       39       69
## Sigma      1.3    3.3    8.6    3.9  2.4  1.03       44      100
## Lp__    -235.9 -212.2 -199.8 -214.3 11.5  1.02       63      289
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

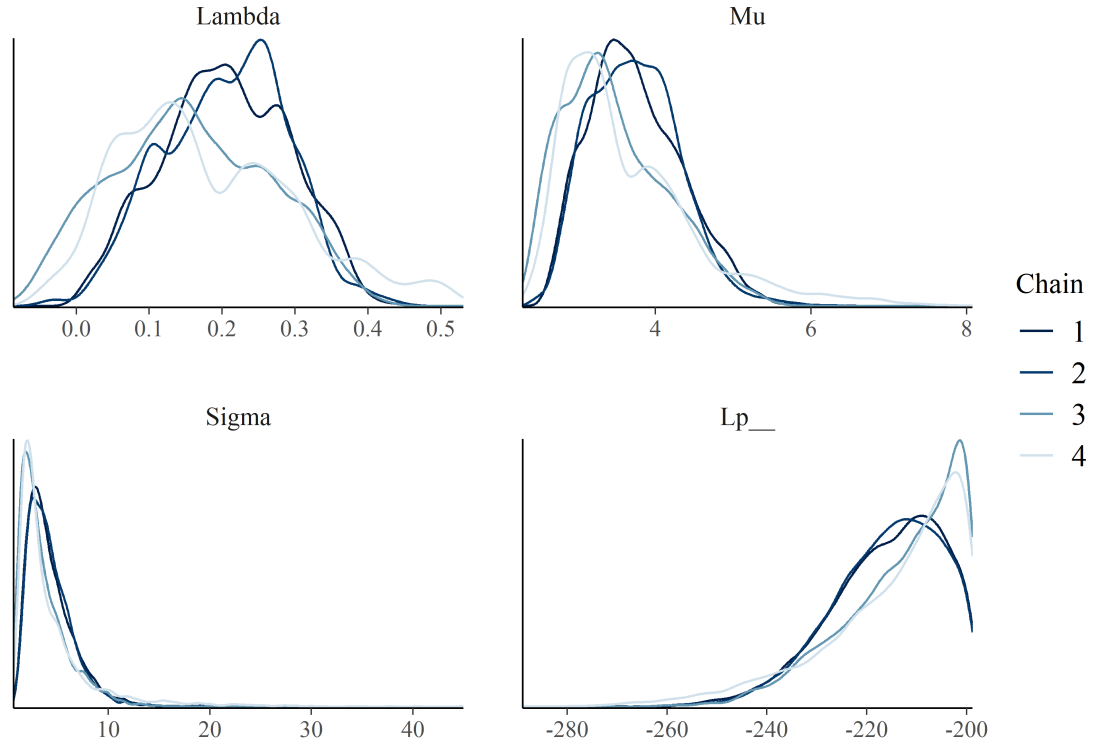
```
sim = sims[5000:10000,,]
mcmc_trace(sim)
```



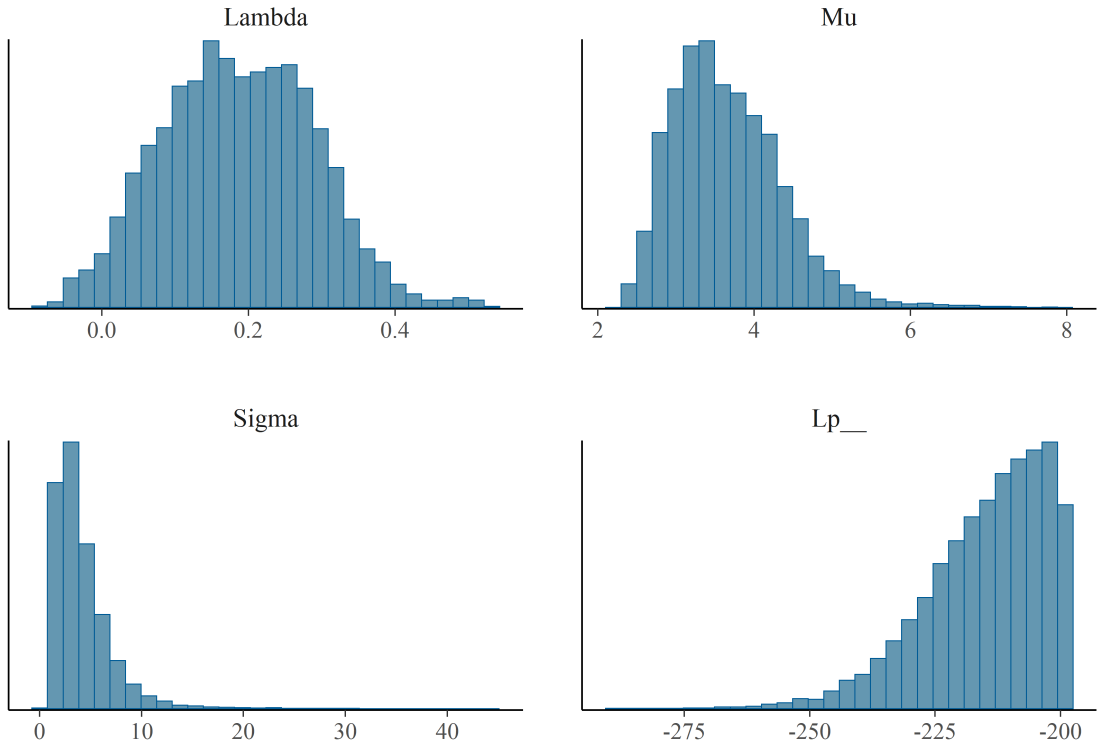
```
mcmc_acf(sim)
```




```
mcmc_dens_overlay(sim)
```



```
mcmc_hist(sim)
```




```
quantile(sim[,,1],c(0.025,0.975))
```

```
##         2.5%        97.5%
## -0.005401647  0.384868231
```

```
quantile(sim[,,2],c(0.025,0.975))
```

```
##     2.5%    97.5%
## 2.598098 5.217696
```

```
quantile(sim[,,3],c(0.025,0.975))
```

```
##      2.5%     97.5%
##  1.173741 11.382321
```

So the 95% Confidence interval is

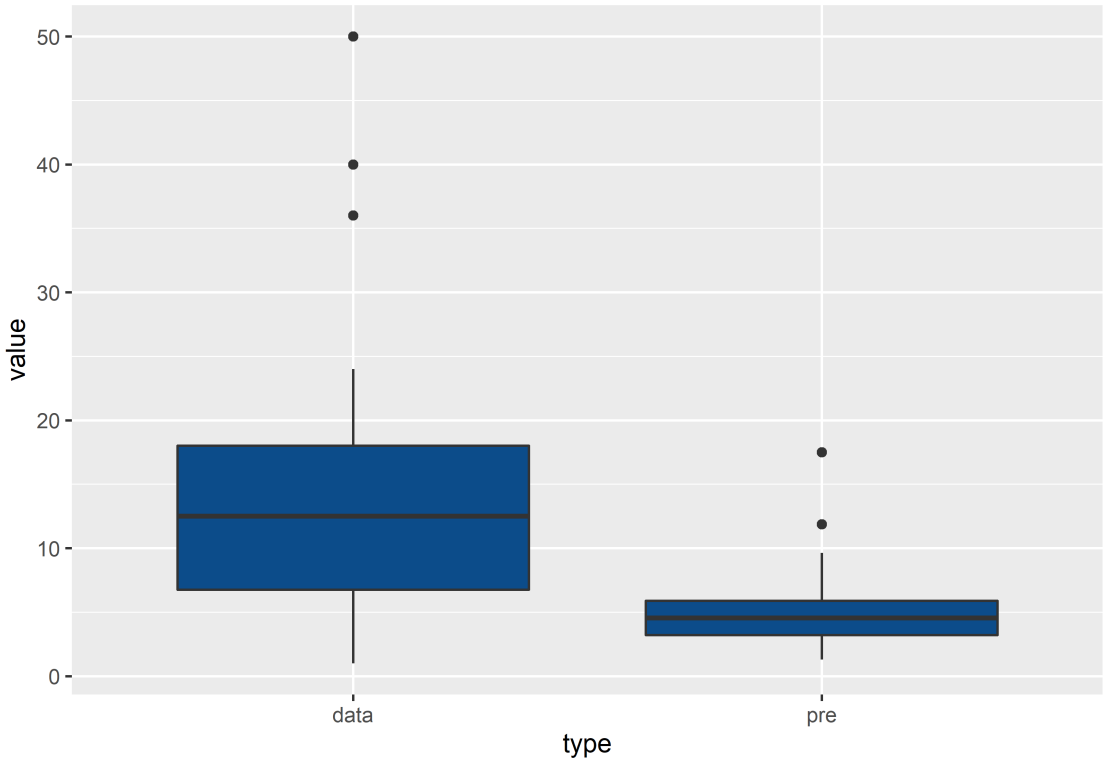
- $\lambda$ : [-0.0668,0.4698]
- $\mu$ : [2.3735,6.1785]
- $\sigma$ : [0.9176,17.7471]

```
ss = n
lambda = sim[,,1]
mu = sim[,,2]
sigma = sim[,,3]
la.sa = sample(x = lambda,size = ss)
mu.sa = sample(x = mu,size = ss)
si.sa = sample(x = sigma,size = ss)
w = rnorm(n = ss,mean = mu.sa,sd = si.sa)
y = array(dim = ss)
```

```
for(i in 1:ss){
  if(abs(la.sa[i])<0.005){y[i] = exp(w[i])}else{
    y[i] = (w[i]*(la.sa[i]+1)^(la.sa[i]))
  }
}
sam = data.frame(cbind(y,w,la.sa,mu.sa,si.sa))
#sam %<>% filter(y>0)
com = data.frame("pre" = sam$y,"data" = data$y) %>% filter(pre >0)
quantile(com$pre,c(0.025,0.975))
```

```
##      2.5%     97.5%
##  1.651724 11.810318
```

```
com = pivot_longer(data = com,cols = colnames(com),names_to = "type", values_to = "value")
com <- com %>%
 filter(value >= 0L & value <= 50L)

ggplot(com) +
 aes(x = type, y = value) +
 geom_boxplot(fill = "#0c4c8a")
```




The 95% Confidence Interval for $\tilde{y}$ is [0.0707,40.4988]

From the boxplot, we can see that the predicted $\tilde{y}$ has obvious lower mean and variance.